



Reference: Statements and Options
SAP Sybase IQ 16.0 SP01

DOCUMENT ID: DC00801-01-1601-01

LAST REVISED: May 2013

Copyright © 2013 by SAP AG or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP AG. The information contained herein may be changed without prior notice.

Some software products marketed by SAP AG and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP AG and its affiliated companies ("SAP Group") for informational purposes only, without representation or warranty of any kind, and SAP Group shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP Group products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and other countries. Please see <http://www.sap.com/corporate-en/legal/copyright/index.epx#trademark> for additional trademark information and notices.

Contents

Audience	1
SQL Statements	3
Common Elements in SQL Syntax	3
Syntax Conventions	4
Statement Applicability Indicators	5
ALLOCATE DESCRIPTOR Statement [ESQL]	5
ALTER DATABASE Statement	7
ALTER DBSPACE Statement	9
ALTER DOMAIN Statement	13
ALTER EVENT Statement	14
ALTER FUNCTION Statement	16
ALTER INDEX Statement	17
ALTER LDAP SERVER Statement	20
ALTER LOGICAL SERVER Statement	22
ALTER LOGIN POLICY Statement	24
Login Policy Options	26
LDAP Login Policy Options	29
Multiplex Login Policy Configuration	30
Logical Server Access Configuration	30
ALTER LS POLICY Statement	32
LS Policy Options	33
ALTER MULTIPLEX RENAME Statement	34
ALTER MULTIPLEX SERVER Statement	35
ALTER PROCEDURE Statement	36
ALTER ROLE Statement	38
ALTER SEQUENCE statement	40
ALTER SERVER Statement	41
ALTER SERVICE Statement	44
ALTER SPATIAL REFERENCE SYSTEM Statement ..	46
ALTER TABLE Statement	52
ALTER TEXT INDEX Statement	66

ALTER TEXT CONFIGURATION Statement	68
ALTER TRIGGER statement	70
ALTER USER Statement	71
ALTER VIEW Statement	75
Identifying and Fixing Invalid Dependent Views	77
BACKUP Statement	78
BEGIN ... END Statement	84
BEGIN PARALLEL IQ ... END PARALLEL IQ Statement	87
BEGIN TRANSACTION Statement [T-SQL]	88
CALL Statement	90
CASE Statement	92
CHECKPOINT Statement	94
CLEAR Statement [Interactive SQL]	94
CLOSE Statement [ESQL] [SP]	95
COMMENT Statement	96
COMMIT Statement	102
CONFIGURE Statement [Interactive SQL]	104
CONNECT Statement [ESQL] [Interactive SQL]	104
CREATE DATABASE Statement	107
CREATE DBSPACE Statement	117
CREATE DOMAIN Statement	121
CREATE EVENT Statement	123
CREATE EXISTING TABLE Statement	129
CREATE EXTERNLOGIN Statement	131
CREATE FUNCTION Statement	133
CREATE FUNCTION Statement (Java UDF) ...	139
CREATE INDEX Statement	142
CREATE LDAP SERVER Statement	150
CREATE LOGICAL SERVER Statement	153
CREATE LOGIN POLICY Statement	155
Login Policy Options	157
LDAP Login Policy Options	160
Multiplex Login Policy Configuration	161

CREATE LS POLICY Statement	162
CREATE MESSAGE Statement [T-SQL]	163
CREATE MULTIPLEX SERVER Statement	164
CREATE PROCEDURE Statement	165
Referencing Temporary Tables Within Procedures	172
CREATE PROCEDURE Statement [T-SQL]	172
CREATE PROCEDURE Statement (External Procedures)	175
CREATE PROCEDURE Statement (Java UDF)	181
CREATE PROCEDURE Statement (Table UDF)	183
CREATE ROLE Statement	187
CREATE SCHEMA Statement	189
CREATE SEQUENCE statement	190
CREATE SERVER Statement	192
CREATE SERVICE Statement	194
CREATE SPATIAL REFERENCE SYSTEM Statement	197
CREATE SPATIAL UNIT OF MEASURE Statement .	204
CREATE TABLE Statement	205
CREATE TEXT CONFIGURATION Statement	222
CREATE TEXT INDEX Statement	223
CREATE TRIGGER statement	224
CREATE USER Statement	230
CREATE VARIABLE Statement	232
CREATE VIEW Statement	234
DEALLOCATE DESCRIPTOR Statement [ESQL]	237
Declaration Section [ESQL]	237
DECLARE Statement	238
DECLARE CURSOR Statement [ESQL] [SP]	240
DECLARE CURSOR Statement [T-SQL]	245
DECLARE LOCAL TEMPORARY TABLE Statement .	246
DELETE Statement	248
DELETE (positioned) Statement [ESQL] [SP]	250

DESCRIBE Statement [ESQL]	251
DISCONNECT Statement [Interactive SQL]	254
DROP Statement	255
DROP CONNECTION Statement	259
DROP DATABASE Statement	260
DROP EXTERNLOGIN Statement	261
DROP LDAP SERVER Statement	262
DROP LOGIN POLICY Statement	263
DROP LOGICAL SERVER Statement	264
DROP LS POLICY Statement	265
DROP MULTIPLEX SERVER Statement	265
DROP ROLE Statement	266
DROP SEQUENCE statement	268
DROP SERVER Statement	269
DROP SERVICE Statement	269
DROP SPATIAL REFERENCE SYSTEM Statement .	270
DROP SPATIAL UNIT OF MEASURE Statement	271
DROP STATEMENT Statement [ESQL]	271
DROP TEXT CONFIGURATION Statement	272
DROP TEXT INDEX Statement	273
DROP TRIGGER statement	274
DROP USER Statement	275
DROP VARIABLE Statement	276
EXECUTE Statement [ESQL]	276
EXECUTE Statement [T-SQL]	278
EXECUTE IMMEDIATE Statement [ESQL] [SP]	279
EXIT Statement [Interactive SQL]	282
FETCH Statement [ESQL] [SP]	283
FOR Statement	286
FOR JSON Statement	289
FORWARD TO Statement	294
FROM Clause	295
GET DESCRIPTOR Statement [ESQL]	303
GOTO Statement [T-SQL]	304
GRANT CHANGE PASSWORD Statement	304

GRANT CONNECT Statement	306
GRANT CREATE Statement	308
GRANT Object-Level Privilege Statement	309
GRANT EXECUTE Statement	310
GRANT INTEGRATED LOGIN Statement	311
GRANT KERBEROS LOGIN Statement	312
GRANT ROLE Statement	312
GRANT SET USER Statement	317
GRANT System Privilege Statement	319
List of All System Privileges	320
GRANT USAGE ON SEQUENCE Statement	323
IF Statement	323
IF Statement [T-SQL]	325
INCLUDE Statement [ESQL]	326
INSERT Statement	327
INSTALL JAVA Statement	336
IQ UTILITIES Statement	338
LEAVE Statement	342
LOAD TABLE Statement	343
Storage Sizes	361
LOCK TABLE Statement	362
LOOP Statement	364
MESSAGE Statement	366
OPEN Statement [ESQL] [SP]	369
OUTPUT Statement [Interactive SQL]	371
PARAMETERS Statement [Interactive SQL]	374
PREPARE Statement [ESQL]	375
PRINT Statement [T-SQL]	378
PUT Statement [ESQL]	379
RAISERROR Statement [T-SQL]	381
READ Statement [Interactive SQL]	382
REFRESH TEXT INDEX Statement	384
RELEASE SAVEPOINT Statement	386
REMOVE Statement	387
RESIGNAL Statement	388

RESTORE DATABASE Statement	389
RESUME Statement	395
RETURN Statement	396
REVOKE CHANGE PASSWORD Statement	398
REVOKE CONNECT Statement	399
REVOKE CREATE Statement	400
REVOKE EXECUTE Statement	401
REVOKE INTEGRATED LOGIN Statement	401
REVOKE KERBEROS LOGIN Statement	402
REVOKE Object-Level Privilege Statement	403
REVOKE ROLE Statement	404
REVOKE SET USER Statement	407
REVOKE System Privilege Statement	408
List of All System Privileges	410
REVOKE USAGE ON SEQUENCE Statement	412
ROLLBACK Statement	413
ROLLBACK TO SAVEPOINT Statement	413
ROLLBACK TRANSACTION Statement [T-SQL]	414
SAVEPOINT Statement	415
SAVE TRANSACTION Statement [T-SQL]	416
SELECT Statement	417
SET Statement [ESQL]	426
SET Statement [T-SQL]	427
SET CONNECTION Statement [ESQL] [Interactive SQL]	429
SET DESCRIPTOR Statement [ESQL]	430
SET OPTION Statement	431
SET OPTION Statement [Interactive SQL]	434
SET SQLCA Statement [ESQL]	435
SETUSER Statement	436
SIGNAL Statement	437
START DATABASE Statement [Interactive SQL]	438
START ENGINE Statement [Interactive SQL]	439
START JAVA Statement	440
STOP DATABASE Statement [Interactive SQL]	441

STOP ENGINE Statement [Interactive SQL]	442
STOP JAVA Statement	442
TRIGGER EVENT Statement	443
TRUNCATE Statement	444
TRUNCATE TEXT INDEX Statement	445
UNION Operation	446
UPDATE Statement	448
UPDATE (positioned) Statement [ESQL] [SP]	451
VALIDATE Statement	453
VALIDATE LDAP SERVER Statement	455
WAITFOR Statement	458
WHENEVER Statement [ESQL]	459
WHILE Statement [T-SQL]	460
Database Options	463
Introduction to Database Options	463
Current Option Settings	464
Scope and Duration of Database Options	465
Temporary Options	466
PUBLIC Options	466
SECURITY Options	466
SYSTEM Options	467
Delete an Option Setting	467
Initial Option Settings	468
Deprecated Database Options	469
General Database Options	469
Data Extraction Options	474
Transact-SQL Compatibility Options	474
Transact-SQL Option Settings for Adaptive Server Enterprise Compatibility	475
Interactive SQL Options	476
Alphabetical List of Options	477
AFFINITY_AUTOEXCLUDE_TIMEOUT Option	477
AGGREGATION_PREFERENCE Option	478

ALLOW_NULLS_BY_DEFAULT Option [TSQL]	479
ALLOW_SNAPSHOT_VERSIONING Option	480
ANSI_CLOSE_CURSORS_ON_ROLLBACK	
Option [TSQL]	480
ANSI_PERMISSIONS Option [TSQL]	481
ANSINULL Option [TSQL]	482
ANSI_SUBSTRING Option [TSQL]	483
ANSI_UPDATE_CONSTRAINTS Option	484
ALLOW_READ_CLIENT_FILE Option	485
ASE_BINARY_DISPLAY Option	486
ASE_FUNCTION_BEHAVIOR Option	487
AUDITING Option [database]	488
BASE_TABLES_IN_RLV_STORE Option	488
BIT_VECTOR_PINNABLE_CACHE_PERCEN	
T Option	489
BLOCKING Option	490
BLOCKING_TIMEOUT Option	490
BT_PREFETCH_MAX_MISS Option	491
BT_PREFETCH_SIZE Option	492
BTREE_PAGE_SPLIT_PAD_PERCENT Option	493
CACHE_AFFINITY_PERCENT Option	494
CACHE_PARTITIONS Option	494
CHAINED Option [TSQL]	495
CHECKPOINT_TIME Option	496
CIS_ROWSET_SIZE Option	497
CLOSE_ON_ENDTRANS Option [TSQL]	497
CONTINUE_AFTER_RAISERROR Option	
[TSQL]	498
CONVERSION_ERROR Option [TSQL]	499
CONVERSION_MODE Option	499
CONVERT_VARCHAR_TO_1242 Option	506
COOPERATIVE_COMMIT_TIMEOUT Option	506
COOPERATIVE_COMMITS Option	507

CREATE_HG_WITH_EXACT_DISTINCTS	508
CREATE_HG_AND_FORCE_PHYSICAL_DEL ETE	509
CURSOR_WINDOW_ROWS Option	510
DATE_FIRST_DAY_OF_WEEK Option	510
DATE_FORMAT Option	512
DATE_ORDER Option	514
DBCC_LOG_PROGRESS Option	514
DBCC_PINNABLE_CACHE_PERCENT Option	515
DEBUG_MESSAGES Option	516
DEDICATED_TASK Option	517
DEFAULT_DBSPACE Option	517
DEFAULT_DISK_STRIPING Option	519
DEFAULT_HAVING_SELECTIVITY_PPM Option	519
DEFAULT_ISQL_ENCODING Option [Interactive SQL]	520
DEFAULT_KB_PER_STRIPE Option	521
DEFAULT_LIKE_MATCH_SELECTIVITY_PPM Option	522
DEFAULT_LIKE_RANGE_SELECTIVITY_PPM Option	523
DEFAULT_PROXY_TABLE_ROW_COUNT Option	523
DEFAULT_TABLE_UDF_ROW_COUNT Option	524
DELAYED_COMMIT_TIMEOUT Option	525
DELAYED_COMMITS Option	525
DISABLE_RI_CHECK Option	526
DIVIDE_BY_ZERO_ERROR Option [TSQL]	526
DQP_ENABLED Option	527
DQP_ENABLED_OVER_NETWORK Option	528
EARLY_PREDICATE_EXECUTION Option	528
ENABLE_ASYNC_IO Option	530

ENABLE_LOB_VARIABLES Option530

EXTENDED_JOIN_SYNTAX Option 531

FLOATING_POINT_ACCUMULATOR Option ...532

FORCE_DROP Option 532

FORCE_NO_SCROLL_CURSORS Option533

FORCE_UPDATABLE_CURSORS Option 534

FP_LOOKUP_SIZE Option535

FP_LOOKUP_SIZE_PPM Option535

FP_NBIT_AUTOSIZE_LIMIT Option 536

FP_NBIT_ENFORCE_LIMITS Option 538

FP_NBIT_IQ15_COMPATIBILITY Option 539

FP_NBIT_LOOKUP_MB Option 540

FP_NBIT_ROLLOVER_MAX_MB Option542

FP_PREDICATE_WORKUNIT_PAGES Option
..... 543

FPL_EXPRESSION_MEMORY_KB Option 543

GARRAY_FILL_FACTOR_PERCENT Option ...544

GARRAY_INSERT_PREFETCH_SIZE Option ..545

GARRAY_PAGE_SPLIT_PAD_PERCENT
Option545

GARRAY_RO_PREFETCH_SIZE Option546

HASH_PINNABLE_CACHE_PERCENT Option
..... 547

HASH_THRASHING_PERCENT Option548

HG_DELETE_METHOD Option 548

HG_SEARCH_RANGE Option 549

HTTP_SESSION_TIMEOUT Option550

IDENTITY_ENFORCE_UNIQUENESS Option . 551

IDENTITY_INSERT Option551

IN_SUBQUERY_PREFERENCE Option 552

INDEX_ADVISOR Option554

INDEX_ADVISOR_MAX_ROWS Option 556

INDEX_PREFERENCE Option557

INFER_SUBQUERY_PREDICATES Option558

IQGOVERN_MAX_PRIORITY Option 559

IQGOVERN_PRIORITY Option	559
IQGOVERN_PRIORITY_TIME Option	560
ISOLATION_LEVEL Option	561
JAVA_LOCATION Option	561
JAVA_VM_OPTIONS Option	562
JOIN_EXPANSION_FACTOR Option	562
JOIN_OPTIMIZATION Option	563
JOIN_PREFERENCE Option	565
JOIN_SIMPLIFICATION_THRESHOLD Option	567
LF_BITMAP_CACHE_KB Option	567
LOAD_ZEROLENGTH_ASNULL Option	568
LOG_CONNECT Option	569
LOG_CURSOR_OPERATIONS Option	570
LOG_DEADLOCKS Option	570
LOGIN_MODE Option	571
LOGIN_PROCEDURE Option	572
MAIN_RESERVED_DBSPACE_MB Option	572
MAX_CARTESIAN_RESULT Option	573
MAX_CLIENT_NUMERIC_PRECISION Option	574
MAX_CLIENT_NUMERIC_SCALE Option	575
MAX_CUBE_RESULT Option	576
MAX_CURSOR_COUNT Option	576
MAX_HASH_ROWS Option	577
MAX_IQ_THREADS_PER_CONNECTION Option	578
MAX_IQ_THREADS_PER_TEAM Option	578
MAX_JOIN_ENUMERATION Option	579
MAX_PARTITIONED_HASH_MB Option	580
MAX_PREFIX_PER_CONTAINS_PHRASE Option	581
MAX_QUERY_PARALLELISM Option	581
MAX_QUERY_TIME Option	582
MAX_STATEMENT_COUNT Option	583

MAX_TEMP_SPACE_PER_CONNECTION	
Option	583
MINIMIZE_STORAGE Option	585
MIN_PASSWORD_LENGTH Option	586
MIN_ROLE_ADMINS Option	586
MONITOR_OUTPUT_DIRECTORY Option	587
MPX_AUTOEXCLUDE_TIMEOUT Option	588
MPX_HEARTBEAT_FREQUENCY Option	588
MPX_IDLE_CONNECTION_TIMEOUT Option	589
MPX_LIVENESS_TIMEOUT Option	589
MPX_MAX_CONNECTION_POOL_SIZE	
Option	590
MPX_MAX_UNUSED_POOL_SIZE Option	591
MPX_WORK_UNIT_TIMEOUT Option	591
NEAREST_CENTURY Option [TSQL]	592
NOEXEC Option	593
NON_ANSI_NULL_VARCHAR Option	593
NON_KEYWORDS Option [TSQL]	594
NOTIFY_MODULUS Option	595
ODBC_DISTINGUISH_CHAR_AND_VARCHA	
R Option	595
ON_CHARSET_CONVERSION_FAILURE	
Option	596
ON_ERROR Option [Interactive SQL]	597
ON_TSQL_ERROR Option [TSQL]	598
OS_FILE_CACHE_BUFFERING Option	599
OS_FILE_CACHE_BUFFERING_TEMPDB	
Option	600
POST_LOGIN_PROCEDURE Option	601
PRECISION Option	602
PREFETCH Option	603
PREFETCH_BUFFER_LIMIT Option	603
PREFETCH_BUFFER_PERCENT Option	604
PREFETCH_GARRAY_PERCENT Option	604

PREFETCH_SORT_PERCENT Option	605
PRESERVE_SOURCE_FORMAT Option [database]	606
QUERY_DETAIL Option	606
QUERY_NAME Option	607
QUERY_PLAN Option	608
QUERY_PLAN_AFTER_RUN Option	608
QUERY_PLAN_AS_HTML Option	609
QUERY_PLAN_AS_HTML_DIRECTORY Option	611
QUERY_PLAN_MIN_TIME Option	612
QUERY_PLAN_TEXT_ACCESS Option	613
QUERY_PLAN_TEXT_CACHING Option	614
QUERY_ROWS_RETURNED_LIMIT Option	615
QUERY_TEMP_SPACE_LIMIT Option	615
QUERY_TIMING Option	616
QUOTED_IDENTIFIER Option [TSQL]	617
RECOVERY_TIME Option	618
RESERVED_KEYWORDS Option	618
RETURN_DATE_TIME_AS_STRING Option	619
REVERT_TO_V15_OPTIMIZER Option	620
ROUND_TO_EVEN Option	620
ROW_COUNT Option	621
RV_AUTO_MERGE_EVAL_INTERVAL Option	622
RV_MAX_ACTIVE_SUBFRAGMENT_COUNT Option	623
RV_MERGE_NODE_MEMSIZE Option	623
RV_MERGE_TABLE_MEMPERCENT Option ...	624
RV_MERGE_TABLE_NUMROWS Option	625
RV_RESERVED_DBSPACE_MB Option	625
SCALE Option	626
SNAPSHOT_VERSIONING Option	627
SIGNIFICANTDIGITSFORDOUBLEEQUALITY Option	628

SORT_COLLATION Option	628
SORT_PINNABLE_CACHE_PERCENT Option	629
SQL_FLAGGER_ERROR_LEVEL Option [TSQL].....	630
SQL_FLAGGER_WARNING_LEVEL Option [TSQL].....	631
STRING_RTRUNCATION Option [TSQL]	632
SUBQUERY_CACHING_PREFERENCE Option	633
SUBQUERY_FLATTENING_PERCENT Option	634
SUBQUERY_FLATTENING_PREFERENCE Option	635
SUBQUERY_PLACEMENT_PREFERENCE Option	636
SUPPRESS_TDS_DEBUGGING Option	637
SWEEPER_THREADS_PERCENT Option	638
TDS_EMPTY_STRING_IS_NULL Option [database].....	638
TEMP_EXTRACT_APPEND Option	639
TEMP_EXTRACT_BINARY Option	640
TEMP_EXTRACT_COLUMN_DELIMITER Option	641
TEMP_EXTRACT_DIRECTORY Option	642
TEMP_EXTRACT_ESCAPE_QUOTES Option	643
TEMP_EXTRACT_NAME _n Options	643
TEMP_EXTRACT_NULL_AS_EMPTY Option	645
TEMP_EXTRACT_NULL_AS_ZERO Option	646
TEMP_EXTRACT_QUOTE Option	647
TEMP_EXTRACT_QUOTES Option	648
TEMP_EXTRACT_QUOTES_ALL Option	648
TEMP_EXTRACT_ROW_DELIMITER Option ...	649

TEMP_EXTRACT_SIZE _n Options	650
TEMP_EXTRACT_SWAP Option	651
TEMP_RESERVED_DBSPACE_MB Option	652
TEMP_SPACE_LIMIT_CHECK Option	652
TEXT_DELETE_METHOD Option	653
TIME_FORMAT Option	654
TIMESTAMP_FORMAT Option	655
TOP_NSORT_CUTOFF_PAGES Option	656
TRIM_PARTIAL_MBC Option	657
TRUSTED_CERTIFICATES_FILE Option	658
TSQL_VARIABLES Option [TSQL]	658
USER_RESOURCE_RESERVATION Option	659
VERIFY_PASSWORD_FUNCTION Option	659
WASH_AREA_BUFFERS_PERCENT Option	662
WAIT_FOR_COMMIT Option	663
WD_DELETE_METHOD Option	663
Index	665

Contents

Audience

This book is for Sybase® IQ users who require reference material for SAP® Sybase® IQ SQL statements and database options.

Reference material for other aspects of SAP Sybase IQ, including language elements, data types, functions, system procedures, and system tables is provided in *Reference: Building Blocks, Tables, and Procedures*. Other books provide more context on how to perform particular tasks. This reference book is the place to look for information such as available SQL syntax, parameters, and options. For command line utility start-up parameters, see the *Utility Guide*.

Audience

SQL Statements

Descriptions of the SQL statements available in SAP Sybase IQ, including some that can be used only from Embedded SQL or Interactive SQL.

Common Elements in SQL Syntax

Language elements that are found in the syntax of many SQL statements.

For more information on the elements described here, see *Identifiers*, *Search Conditions*, *Expressions*, and *Strings* in *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements*.

- column-name – an identifier that represents the name of a column.
- condition – an expression that evaluates to TRUE, FALSE, or UNKNOWN.
- connection-name – a string representing the name of an active connection.
- data-type – a storage data type.
- expression – an expression.
- filename – a string containing a file name.
- host-variable – a C language variable, declared as a host variable, preceded by a colon.
- indicator-variable – a second host variable of type `short int` immediately following a normal host variable. An indicator variable must also be preceded by a colon. Indicator variables are used to pass NULL values to and from the database.
- number – any sequence of digits followed by an optional decimal part and preceded by an optional negative sign. Optionally, the number can be followed by an ‘e’ and then an exponent. For example,

```
42
-4.038
.001
3.4e10
1e-10
```

- owner – an identifier representing the user ID who owns a database object.
- role-name – an identifier representing the role name of a foreign key.
- savepoint-name – an identifier that represents the name of a savepoint.
- search-condition – a condition that evaluates to TRUE, FALSE, or UNKNOWN.
- special-value – one of the special values described in *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements > Special Values*.
- statement-label – an identifier that represents the label of a loop or compound statement.
- table-list – a list of table names, which might include correlation names. For more information, see *FROM clause*.

SQL Statements

- `table-name` – an identifier that represents the name of a table.
- `userid` – an identifier representing a user name. The user ID is not case-sensitive and is unaffected by the setting of the `CASE RESPECT` property of the database.
- `variable-name` – an identifier that represents a variable name.

See also

- *FROM Clause* on page 295

Syntax Conventions

Conventions used in the SQL syntax descriptions.

- **Keywords** – all SQL keywords appear in UPPERCASE; however, SQL keywords are case-insensitive, so you can type keywords in any case. For example, `SELECT` is the same as `Select`, which is the same as `select`.
- **Placeholders** – items that must be replaced with appropriate identifiers or expressions are shown in *italics*.
- **Continuation** – lines beginning with an ellipsis (`...`) are a continuation from the previous line.

- **Optional portions** – optional portions of a statement are enclosed by square brackets. For example:

```
RELEASE SAVEPOINT [ savepoint-name ]
```

This example indicates that the *savepoint-name* is optional. Do not type the square brackets.

- **Repeating items** – lists of repeating items are shown with an element of the list followed by an ellipsis. One or more list elements are allowed. When more than one is specified, they must be separated by commas if indicated as such. For example:

```
UNIQUE ( column-name [ , ... ] )
```

The example indicates that you can specify *column-name* more than once, separated by commas. Do not type the square brackets.

- **Alternatives** – when one option must be chosen, the alternatives are enclosed in curly braces. For example:

```
[ QUOTES { ON | OFF } ]
```

The example indicates that if you choose the `QUOTES` option, you must provide one of `ON` or `OFF`. Do not type the braces.

- **One or more options** – if you choose more than one, separate your choices by commas. For example:

```
{ CONNECT, DBA, RESOURCE }
```

Statement Applicability Indicators

Some statement titles are followed by an indicator in square brackets that shows where the statement can be used.

These indicators are as follows:

- [ESQL] – the statement is for use in Embedded SQL.
- [Interactive SQL] – the statement is for use only in Interactive SQL (**dbisql**).
- [SP] – the statement is for use in stored procedures or batches.
- [T-SQL] – the statement is implemented for compatibility with Adaptive Server® Enterprise. In some cases, the statement cannot be used in stored procedures that are not Transact-SQL format. In other cases, there is an alternative statement that is closer to the ISO/ANSI SQL standard that is recommended unless Transact-SQL compatibility is an issue.

If two sets of brackets are used, the statement can be used in both environments. For example, [ESQL] [SP] means a statement can be used either in Embedded SQL or in stored procedures.

ALLOCATE DESCRIPTOR Statement [ESQL]

Allocates space for a SQL descriptor area (SQLDA).

Syntax

```
ALLOCATE DESCRIPTOR descriptor-name
... [ WITH MAX { integer | host-variable } ]
```

Parameters

- **WITH MAX** – lets you specify the number of variables within the descriptor area. The default size is 1.

Examples

- **Example 1** – this sample program includes an example of **ALLOCATE DESCRIPTOR** statement usage:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

EXEC SQL INCLUDE SQLCA;

#include <sqldef.h>

EXEC SQL BEGIN DECLARE SECTION;
```

SQL Statements

```
int      x;
short   type;
int      numcols;
char     string[100];
a_sql_statement_number stmt = 0;
EXEC SQL END DECLARE SECTION;

int main(int argc, char * argv[])
{
    struct sqllda *      sqllda1;

    if( !db_init( &sqlca ) ) {
        return 1;
    }

    db_string_connect(&sqlca, "UID=dba;PWD=sql;DBF=d:\\IQ-16_0\\
\\sample.db");

    EXEC SQL ALLOCATE DESCRIPTOR sqllda1 WITH MAX 25;

    EXEC SQL PREPARE :stmt FROM
        'select * from Employees';
    EXEC SQL DECLARE curs CURSOR FOR :stmt;
    EXEC SQL OPEN curs;

    EXEC SQL DESCRIBE :stmt into sqllda1;
    EXEC SQL GET DESCRIPTOR sqllda1 :numcols=COUNT;
        // how many columns?
    if( numcols > 25 ) {
        // reallocate if necessary
        EXEC SQL DEALLOCATE DESCRIPTOR sqllda1;
        EXEC SQL ALLOCATE DESCRIPTOR sqllda1
            WITH MAX :numcols;
    }
    type = DT_STRING; // change the type to string
    EXEC SQL SET DESCRIPTOR sqllda1 VALUE 2 TYPE = :type;
    fill_sqllda( sqllda1 ); // allocate space for the variables

    EXEC SQL FETCH ABSOLUTE 1 curs USING DESCRIPTOR sqllda1;
    EXEC SQL GET DESCRIPTOR sqllda1 VALUE 2 :string = DATA;

    printf("name = %s", string );

    EXEC SQL DEALLOCATE DESCRIPTOR sqllda1;
    EXEC SQL CLOSE curs;
    EXEC SQL DROP STATEMENT :stmt;

    db_string_disconnect( &sqlca, "" );
    db_fini( &sqlca );

    return 0;
}
```

Usage

You must declare the following in your C code prior to using this statement:


```
struct sqllda * descriptor_name
```

You must still call `fill_sqllda` to allocate space for the actual data items before doing a fetch or any statement that accesses the data within a descriptor area.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Open Client/Open Server.

See also

- *DEALLOCATE DESCRIPTOR Statement [ESQL]* on page 237

ALTER DATABASE Statement

Upgrades a database created with a previous version of the software, adds or removes jConnect™ for JDBC™ support, or defines management of system procedure execution. Run this statement with DBISQL Interactive SQL.

Syntax

```
ALTER DATABASE UPGRADE
  [ PROCEDURE ON ]
  [ JCONNECT { ON | OFF } ]
  [ RESTART { ON | OFF } ]
  [ SYSTEM PROCEDURE AS DEFINER {ON | OFF} ]
```

Parameters

- **PROCEDURE** – drops and re-creates all dbo- and sys-owned procedures in the database.
- **JCONNECT** – to allow the SAP Sybase IQ jConnect JDBC driver to access system catalog information, you must specify ON. This installs jConnect system tables and procedures. To exclude the jConnect system objects, specify OFF. You can still use JDBC, as long as you do not access system catalog information. The default is to include jConnect support (JCONNECT ON).
- **RESTART** – when ON (default) is specified and the AutoStop connection parameter is set to NO, the database restarts after it is upgraded. Otherwise, the database is stopped after an upgrade.
- **SYSTEM PROCEDURE AS DEFINER** – defines whether a privileged system procedure runs with the privileges of the invoker (the person executing the procedure) or the definer (the owner of the procedure).
 - **OFF** – all privileged system procedures execute with the privileges of the invoker. Use `sp_proc_priv()` to identify the system privileges required to run a system procedure.

- **ON (default), or not specified** –
 - when upgrading a pre-16.0 database, pre-16.0 privileged system procedures execute with the privileges of the definer and 16.0 or later privileged system procedures execute with the privileges of the invoker.
 - when upgrading a database that is version 16.0 or later, the default is the behavior of the database being upgraded.

Note: Changing the execution model after upgrade may result in loss of functionality on custom stored procedures and applications that explicitly grant EXECUTE privilege on system procedures. It may also impact the ability to run system procedures. See *Reference: Building Blocks, Tables, and Procedures > System Procedures*.

Examples

- **Example 1** – disables jConnect support:

```
ALTER DATABASE UPGRADE JCONNECT OFF
```

Usage

The **ALTER DATABASE** statement upgrades databases created with earlier versions of the software. This applies to maintenance releases as well as major releases.

When you upgrade a database, SAP Sybase IQ makes these changes:

- Upgrades the system tables to the current version.
- Adds any new database options.
- Enables new features in the current version.

You can also use **ALTER DATABASE UPGRADE** simply to add jConnect features, if the database was created with the current version of the software.

Note:

- See the *Installation and Configuration Guide* for backup recommendations before you upgrade.
- Be sure to start the server in a way that restricts user connections before you run **ALTER DATABASE UPGRADE**. For instructions and other upgrade caveats, see the *Migration* guide for your platform.
- Use the **iqunload** utility to upgrade databases created in versions earlier than 15.0. See the *Migration* guide for your platform.

After using **ALTER DATABASE UPGRADE**, shut down the database.

Side effects:

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the ALTER DATABASE system privilege.

See also

- *CREATE DATABASE Statement* on page 107

ALTER DBSPACE Statement

Changes the read/write mode, changes the size, or extends an existing dbspace.

Syntax

```
ALTER DBSPACE dbspace-name
  { ADD new-file-spec [, new-file-spec ... ]
  | DROP FILE logical-file-name [, FILE logical-file-name ... ]
  | RENAME TO newname | RENAME 'new-file-pathname'
  | READONLY | READWRITE
  | ONLINE | OFFLINE
  | STRIPING{ ON | OFF }
  | STRIPESIZEKB size-in-KB
ALTER FILE file-name
  { READONLY | [ FORCE ] READWRITE }
  | SIZE file-size [ KB | MB | GB | TB ]
  | ADD file-size [ KB | MB | GB | TB | PAGES ] }
RENAME PATH 'new-file-pathname'
RENAME TO newname

new-file-spec:
  FILE logical-file-name 'file-path' iq-file-opts

iq-file-opts:
  [ [ SIZE ] file-size ]
  ...[ KB | MB | GB | TB ] ]
  [ RESERVE reserve-size [ KB | MB | GB | TB ] ]
```

Parameters

- **ADD** – adds one or more files to the specified dbspace. The dbfile name and the physical file path are required for each file and must be unique. You can add files to IQ main, IQ shared temporary, or IQ temporary dbspaces. You may add a file to a read-only dbspace, but the dbspace remains read-only. You can add files to multiplex shared temporary dbspaces only in read-only mode (the default for ADD FILE).

SQL Statements

A catalog dbspace may contain only one file, so **ADD FILE** may not be used on catalog dbspaces.

For an RLV dbspace, use **ADD FILE** on simplex servers only. You cannot add a file to a multiplex RLV dbspace.

When used in the **ALTER FILE** clause, extends the size of the file in units of pages, kilobytes (KB), megabytes (MB), gigabytes (GB), or terabytes (TB). The default is MB. You can **ADD** only if the free list (an allocation map) has sufficient room and if the dbspace has sufficient reserved space.

- **DROP FILE** – removes the specified file from an IQ dbspace. The file must be empty. You cannot drop the last file from the specified dbspace. Instead use **DROP DBSPACE** if the dbspace contains only one file. **Rename TO** clause—Renames the *dbspace-name* to a new name. The new name must be unique in the database. You cannot rename `IQ_SYSTEM_MAIN`, `IQ_SYSTEM_MSG`, `IQ_SYSTEM_TEMP`, `IQ_SHARED_TEMP`, or `SYSTEM`.
- **RENAME TO** – when used with the **DROP FILE** clause, renames the pathname of the dbspace that contains a single file. It is semantically equivalent to the **RENAME PATH** clause. An error is returned if the dbspace contains more than one file.

When used with the **ALTER FILE** clause, renames the specified file's logical name to a new name. The new name must be unique in the database.

- **READONLY** – when used with the **DROP** clause, changes any dbspace except `IQ_SYSTEM_MAIN`, `IQ_SYSTEM_TEMP`, `IQ_SYSTEM_MSG`, `IQ_SHARED_TEMP`, and `SYSTEM` to read-only. Disallows DML modifications to any object currently assigned to the dbspace. Can only be used for dbspaces in the IQ main store.

When used with the **ALTER FILE** clause, changes the specified file to read-only. The file must be associated with an IQ main dbspace. You cannot change files in `IQ_SHARED_TEMP` to **READONLY** status.

- **READWRITE** – when used with the **DROP FILE** clause, changes the dbspace to read-write. The dbspace must be online. Can only be used for dbspaces in the IQ main store.

When used with the **ALTER FILE** clause, changes specified IQ main or temporary store dfile to read-write. The file must be associated with an IQ main or temporary dbspace.

- **ONLINE** – puts an offline dbspace and all associated files online. Can only be used for dbspaces in the IQ main store.
- **OFFLINE** – puts an online read-only dbspace and all associated files offline. (Returns an error if the dbspace is read-write, offline already, or not of the IQ main store.) Can only be used for dbspaces in the IQ main store.
- **STRIPING** – changes the disk striping on the dbspace as specified. When disk striping is set **ON**, data is allocated from each file within the dbspace in a round-robin fashion. For

example, the first database page written goes to the first file, the second page written goes to the next file within given dbspace, and so on. Read-only dbspaces are skipped.

- **STRIPESIZEKB** – specifies the number of kilobytes (KB) to write to each file before the disk striping algorithm moves to the next stripe for the specified dbspace.
- **FORCE READWRITE** – when used with the ALTER FILE clause, changes the status of the specified shared temporary store dbfile to read-write, although there may be known file status problems on secondary nodes. The file may be associated with an IQ main, shared temporary, or temporary dbspace, but because new dbfiles in IQ_SYSTEM_MAIN and user main are created read-write, this clause only affects shared temporary dbspaces.
- **SIZE** – specifies the new size of the file in units of kilobytes (KB), megabytes (MB), gigabytes (GB), or terabytes (TB). The default is megabytes. You can increase the size of the dbspace only if the free list (an allocation map) has sufficient room and if the dbspace has sufficient reserved space. You can decrease the size of the dbspace only if the portion to be truncated is not in use.
- **RENAME PATH** – when used with the ALTER FILE clause, renames the file pathname associated with the specified file. This clause merely associates the file with the new file path instead of the old path. The clause does not actually change the operating system file name. You must change the file name through your operating system. The dbspace must be offline to rename the file path. The new path is used when the dbspace is altered online or when the database is restarted.

You may not rename the path of a file in IQ_SYSTEM_MAIN, because if the new path were not accessible, the database would be unable to start. If you need to rename the path of a file in IQ_SYSTEM_MAIN, make the file read-only, empty the file, drop the file, and add the file again with the new file path name. Enclose the physical file path to the dbfile in single quotes.

Examples

- **Example 1** – changes the mode of a dbspace called DspHist to READONLY:

```
ALTER DBSPACE DspHist READONLY
```

- **Example 2** – adds 500MB to the dbspace DspHist by adding the file FileHist3 of size 500MB:

```
ALTER DBSPACE DspHist
ALTER FILE FileHist3 ADD 500MB
```

- **Example 3** – on a UNIX system, adds two 500MB files to the dbspace DspHist:

```
ALTER DBSPACE DspHist ADD
FILE FileHist3 '/History1/data/file3' SIZE 500MB,
FILE FileHist4 '/History1/data/file4' SIZE 500
```

- **Example 4** – increases the size of the dbspace IQ_SYSTEM_TEMP by 2GB:

SQL Statements

```
ALTER DBSPACE IQ_SYSTEM_TEMP ADD 2 GB
```

- **Example 5** – removes two files from dbspace DspHist. Both files must be empty:

```
ALTER DBSPACE DspHist  
DROP FILE FileHist2, FILE FileHist4
```

- **Example 6** – increases the size of the dbspace IQ_SYSTEM_MAIN by 1000 pages. (ADD clause defaults to pages):

```
ALTER DBSPACE IQ_SYSTEM_MAIN ADD 1000
```

Usage

ALTER DBSPACE changes the read-write mode, changes the online/offline state, alters the file size, renames the dbspace name, file logical name or file path, or sets the dbspace striping parameters. For details about existing dbspaces, run **sp_iqdbspace** procedure, **sp_iqdbspaceinfo** procedure, **sp_iqfile** procedure, **sp_iqdbspaceobjectinfo**, and **sp_iqobjectinfo**. Dbspace and dfile names are always case-insensitive. The physical file paths are case-sensitive, if the database is CASE RESPECT and the operating system supports case-sensitive files. Otherwise, the file paths are case-insensitive.

Enclose dbspace and dfile names either in no quotes or in double quotes.

In Windows, if you specify a path, any backslash characters (\) must be doubled if they are followed by an n or an x. This prevents them being interpreted as a newline character (\n) or as a hexadecimal number (\x), according to the rules for strings in SQL. It is safer to always double the backslash.

Side effects:

- Automatic commit
- Automatic checkpoint
- A mode change to READONLY causes immediate relocation of the internal database structures on the dbspace to one of the read-write dbspaces.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the MANAGE ANY DBSPACE system privilege.

See also

- *CREATE DATABASE Statement* on page 107
- *CREATE DBSPACE Statement* on page 117
- *DROP Statement* on page 255

ALTER DOMAIN Statement

Renames a user-defined domain or data type. Does not rename Java types.

Syntax

```
ALTER { DOMAIN | DATATYPE } user-type  
  RENAME new-name
```

Parameters

- **new-name** – an identifier representing the new domain name.
- **user-type** – user-defined data type of the domain being renamed.

Examples

- **Example 1** – renames the Address domain to MailingAddress:

```
ALTER DOMAIN Address RENAME MailingAddress
```

Usage

The **ALTER DOMAIN** statement updates the name of the user-defined domain or data type in the SYSUSERTYPE system table.

You must recreate any procedures, views or events that reference the user-defined domain or data type, or else they will continue to reference the former name.

Side effects:

- Automatic commit

Permissions

Must be the database user who created the domain or requires the ALTER DATATYPE or ALTER ANY OBJECT system privilege.

See also

- *CREATE DOMAIN Statement* on page 121

ALTER EVENT Statement

Changes the definition of an event or its associated handler for automating predefined actions. Also alters the definition of scheduled actions.

Syntax

```
ALTER EVENT event-name
  [ DELETE TYPE | TYPE event-type ]
  { WHERE { trigger-condition | NULL }
  | { ADD | [ MODIFY ] | DELETE } SCHEDULE schedule-spec }
  [ ENABLE | DISABLE ]
  [ [ MODIFY ] HANDLER compound-statement | DELETE HANDLER }
```

event-type:

```
BackupEnd
| "Connect"
| ConnectFailed
| DatabaseStart
| DBDiskSpace
| "Disconnect"
| GlobalAutoincrement
| GrowDB
| GrowLog
| GrowTemp
| IQMainDBSpaceFree
| IQTempDBSpaceFree
| LogDiskSpace
| "RAISERROR"
| ServerIdle
| TempDiskSpace
```

trigger-condition:

```
event_condition( condition-name )
{ =
| <
| >
| !=
| <=
| >= } value
```

schedule-spec:

```
[ schedule-name ]
{ START TIME start-time | BETWEEN start-time AND end-time }
[ EVERY period { HOURS | MINUTES | SECONDS } ]
[ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]
[ START DATE start-date ]
```

Parameters

- **DELETE TYPE** – removes an association of the event with an event type.

- **ADD | MODIFY | DELETE SCHEDULE** – changes the definition of a schedule. Only one schedule can be altered in any one ALTER EVENT statement.
- **WHERE** – determines the condition under which an event is fired. The WHERE NULL option deletes a condition.

Note: For other parameter descriptions, see the CREATE EVENT Statement.

Examples

- **Example 1** – lists event names by querying the system table SYSEVENT:

```
SELECT event_id, event_name FROM SYS.SYSEVENT
```

- **Example 2** – lists schedule names by querying the system table SYSSCHEDULE:

```
SELECT event_id, sched_name FROM SYS.SYSSCHEDULE
```

Usage

ALTER EVENT lets you alter an event definition created with CREATE EVENT. Possible uses include:

- Change an event handler during development.
- Define and test an event handler without a trigger condition or schedule during a development phase, and then add the conditions for execution using ALTER EVENT once the event handler is completed.
- Disable an event handler temporarily by disabling the event.

When you alter an event using ALTER EVENT, specify the event name and, optionally, the schedule name.

Each event has a unique event ID. Use the event_id columns of SYSEVENT and SYSSCHEDULE to match the event to the associated schedule.

Side effects:

- Automatic commit

Permissions

Requires one of:

- MANAGE ANY EVENT system privilege.
- ALTER ANY OBJECT system privilege.

See also

- *BEGIN ... END Statement* on page 84
- *CREATE EVENT Statement* on page 123

ALTER FUNCTION Statement

Modifies an existing function. Include the entire modified function in the **ALTER FUNCTION** statement.

Syntax

Syntax 1

```
ALTER FUNCTION [ owner.] function-name function-definition
```

function-definition:

```
CREATE FUNCTION syntax
```

Syntax 2

```
ALTER FUNCTION [ owner.] function-name  
SET HIDDEN
```

Syntax 3

```
ALTER FUNCTION [ owner.] function-name  
RECOMPILE
```

Parameters

- **SET HIDDEN** – to scramble the definition of the associated function and cause it to become unreadable. The function can be unloaded and reloaded into other databases.

Warning! The SET HIDDEN clause setting is irreversible. If you need the original source again, you must maintain it outside the database.

- **RECOMPILE** – to recompile a user-defined function. When you recompile a function, the definition stored in the catalog is re-parsed and the syntax is verified. The preserved source for a function is not changed by recompiling. When you recompile a function, the definitions scrambled by the SET HIDDEN clause remain scrambled and unreadable.

Usage

- **Syntax 1** – identical in syntax to the **CREATE FUNCTION** statement except for the first word. Either version of the **CREATE FUNCTION** statement can be altered.

Existing permissions on the function are maintained and do not have to be reassigned. If a **DROP FUNCTION** and **CREATE FUNCTION** were carried out, execute permissions must be reassigned.

Side effects:

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.

Permissions

Alter a Watcom SQL or Transact-SQL function – Requires one of:

- ALTER ANY PROCEDURE system privilege.
- ALTER ANY OBJECT system privilege.
- You own the function.

Alter an external C/C++ Scalar or Aggregate, or external Java function – Requires one of:

- Requires CREATE EXTERNAL REFERENCE system privilege.
- Also requires one of:
 - ALTER ANY PROCEDURE system privilege.
 - ALTER ANY OBJECT system privilege.
 - You own the function.

See also

- *ALTER PROCEDURE Statement* on page 36
- *CREATE FUNCTION Statement* on page 133
- *DROP Statement* on page 255

ALTER INDEX Statement

Renames indexes in base or global temporary tables, foreign key role names of indexes and foreign keys explicitly created by a user, or changes the clustered nature of an index on a catalog store table. You cannot rename indexes created to enforce key constraints.

Syntax

```
ALTER { INDEX index-name
      | [ INDEX ] FOREIGN KEY role-name
      | [ INDEX ] PRIMARY KEY
      | ON [owner.]table-name { rename-clause | move-clause | cluster-clause}

rename-clause:
    RENAME TO | AS new-name

move-clause:
    MOVE TO dbspace-name

cluster-clause:
    CLUSTERED | NONCLUSTERED
```

Parameters

- **ON** – specifies the name of the table that contains the index or foreign key to rename.
- **RENAME** – specifies the new name of the index or foreign key role.
- **MOVE** – moves the specified index, unique constraint, foreign key, or primary key to the specified dbspace. For unique constraint or foreign key, you must specify its unique index name.
- **cluster-clause** – specifies whether the index should be changed to **CLUSTERED** or **NONCLUSTERED**. Applies to catalog store tables only and only one index on a table can be clustered.

Examples

- **Example 1** – move the primary key, HG for c5, from dbspace Dsp4 to Dsp8:

```
CREATE TABLE foo (  
    c1 INT IN Dsp1,  
    c2 VARCHAR(20),  
    c3 CLOB IN Dsp2,  
    c4 DATE,  
    c5 BIGINT,  
    PRIMARY KEY (c5) IN Dsp4) IN Dsp3;  
CREATE DATE INDEX c4_date ON foo(c4) IN Dsp5;  
ALTER INDEX PRIMARY KEY ON foo MOVE TO Dsp8;
```

- **Example 2** – move DATE index from Dsp5 to Dsp9:

```
ALTER INDEX c4_date ON foo MOVE TO Dsp9
```

- **Example 3** – rename an index COL1_HG_OLD in the table jal.mytable to COL1_HG_NEW:

```
ALTER INDEX COL1_HG_OLD ON jal.mytable  
RENAME AS COL1_HG_NEW
```

- **Example 4** – rename a foreign key role name ky_dept_id in table dba.Employees to emp_dept_id:

```
ALTER INDEX FOREIGN KEY ky_dept_id  
ON dba.Employees  
RENAME TO emp_dept_id
```

Usage

You must have **CREATE** privilege on the new dbspace and be the table owner or have the **MANAGE ANY DBSPACE** system privilege.

Note: Attempts to alter an index in a local temporary table return the error `index not found`. Attempts to alter a nonuser-created index, such as a default index (FP), return the

error Cannot alter index. Only indexes in base tables or global temporary tables with an owner type of USER can be altered.

Side effects:

- Automatic commit. Clears the Results tab in the Results pane in Interactive SQL. Closes all cursors for the current connection.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

move-clause for materialized view requires one of:

- MANAGE ANY DBSPACE system privilege.
- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- You own the materialized view along with one of:
 - CREATE ANY OBJECT system privilege.
 - CREATE privilege on the target dbspace.

move-clause for all other indexes requires one of:

- MANAGE ANY DBSPACE system privilege.
- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- You own the underlying table or have REFERENCE privilege on the table along with one of:
 - CREATE ANY OBJECT system privilege.
 - CREATE privilege on the target dbspace.

cluster-clause for materialized view requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- You own the materialized view.

cluster-clause for all other indexes, requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- REFERENCE privilege on the table.
- You own the table.

All other clauses require one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- REFERENCE privilege on the table.
- You own the underlying table.

See also

- *ALTER TABLE Statement* on page 52
- *CREATE INDEX Statement* on page 142
- *CREATE TABLE Statement* on page 205

ALTER LDAP SERVER Statement

Any changes to an LDAP server configuration object are applied on subsequent connections. Any connection already started when the change is applied does not immediately reflect the change.

Syntax

```
ALTER LDAP SERVER ldapua-server-name
  { ldapua-server-attrs
    | [ WITH ( SUSPEND | ACTIVATE | REFRESH ) ] }

ldapua-server-attrs:
  SEARCH DN
    URL { 'URL_string' | NULL }
    | ACCESS ACCOUNT { 'DN_string' | NULL }
    | IDENTIFIED BY ( 'password' | NULL }
    | IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
  | AUTHENTICATION URL { 'URL_string' | NULL }
  | CONNECTION TIMEOUT timeout_value
  | CONNECTION RETRIES retry_value
  | TLS { ON | OFF }
```

Parameters

- **URL** – identifies the host (by name or by IP address), port number, and the search to be performed for the DN lookup for a given user ID. This value is validated for correct LDAP URL syntax before it is stored in the `ISYSLDAPSERVER` system table. The maximum size for this string is 1024 bytes.
- **ACCESS ACCOUNT** – user created in the LDAP server for use by SAP Sybase IQ, not a user within SAP Sybase IQ. The distinguished name (DN) for this user is used to connect to the LDAP server. This user has permissions within the LDAP server to search for DN's by user ID in the locations specified by the SEARCH DN URL. The maximum size for this string is 1024 bytes.

- **IDENTIFIED BY** – provides the password associated with the ACCESS ACCOUNT user. The password is stored using symmetric encryption on disk. Use the value NULL to clear the password and set it to none. The maximum size of a clear text password is 255 bytes.
- **IDENTIFIED BY ENCRYPTED** – configures the password associated with the ACCESS ACCOUNT distinguished name in an encrypted format. The binary value is the encrypted password and is stored on disk as is. Use the value NULL to clear the password and set it to none. The maximum size of the binary is 289 bytes. The encrypted key should be a valid varbinary value. Do not enclose the encrypted key in quotation marks.
- **AUTHENTICATION URL** – identifies the host (by name or IP address) and the port number of the LDAP server to use for authentication of the user. This is the value defined for URL_string and is validated for correct LDAP URL syntax before it is stored in ISYSLDAPSERVER system table. The DN of the user obtained from a prior DN search and the user password bind a new connection to the authentication URL. A successful connection to the LDAP server is considered proof of the identity of the connecting user. The maximum size for this string is 1024 bytes.
- **CONNECTION TIMEOUT** – specifies the connection timeout from SAP Sybase IQ to the LDAP server for both DN searches and authentication. This value is in milliseconds, with a default value of 10 seconds.
- **CONNECTION RETRIES** – specifies the number of retries on connections from SAP Sybase IQ to the LDAP server for both DN searches and authentication. The valid range of values is 1– 60, with a default value of 3.
- **TLS** – defines whether the TLS or Secure LDAP protocol is used for connections to the LDAP server for both DN searches and authentication. When set to ON, the TLS protocol is used and the URL would begin with "ldaps://". When set to OFF (or not specified), Secure LDAP protocol is used and the URL begins with "ldaps://". When using the TLS protocol, specify the database security option TRUSTED_CERTIFICATES_FILE with a file name containing the certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server.
- **WITH ACTIVATE** – activates the LDAP server configuration object for immediate use upon creation. This permits the definition and activation of LDAP User Authentication in one statement. The LDAP server configuration object state changes to READY when WITH ACTIVATE is used.

Examples

- **Example 1** – suspends the LDAP server configuration object named apps_primary:

```
ALTER LDAP SERVER apps_primary SUSPEND
```
- **Example 2** – changes the LDAP server configuration object named apps_primary to use a different URL for authentication on host fairfax, sets the port number to 1066,

sets the number of connection retries to 10, and finally activates the LDAP server configuration object:

```
ALTER LDAP SERVER apps_primary
AUTHENTICATION URL 'ldap://my_LDAPserver:1066/'
CONNECTION RETRIES 10
WITH ACTIVATE
```

Usage

In addition to resetting LDAP server configuration object values for attributes, the **ALTER LDAP SERVER** statement allows an administrator to make manual adjustments to a server's state and behavior by putting the LDAP server configuration object in maintenance mode and returning it to service from maintenance mode.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the **MANAGE ANY LDAP SERVER** system privilege.

ALTER LOGICAL SERVER Statement

Modifies configuration for the existing user-defined logical server in the database. This statement enforces consistent shared system temporary store settings across physical nodes shared by logical servers.

Syntax

```
ALTER LOGICAL SERVER logical-server-name
  { alter-ls-clause } [ WITH STOP SERVER ]

alter-ls-clause:
  { ADD MEMBERSHIP '(' { ls-member, ... } ')'
  | DROP MEMBERSHIP '(' { ls-member, ... } ')'
  | POLICY policy-name }

ls-member:
  FOR LOGICAL COORDINATOR | mpx-server-name
```

Parameters

- **logical-server-name** – refers to an existing user-defined logical server name.
- **WITH STOP SERVER** – automatically shuts down all servers in the logical server when the **TEMP_DATA_IN_SHARED_TEMP** database option is changed directly or indirectly.

Applies to

Multiplex only.

Examples

- **Example 1** – alters a user-defined logical server by adding multiplex nodes `n1` and `n2` to logical server `ls1`:

```
ALTER LOGICAL SERVER ls1 ADD MEMBERSHIP (n1, n2)
```

- **Example 2** – adds logical membership of COORDINATOR and drop a named membership of the current coordinator node `n1` from logical server `ls1`:

```
ALTER LOGICAL SERVER ls1 ADD MEMBERSHIP (FOR LOGICAL COORDINATOR)
ALTER LOGICAL SERVER ls1 DROP MEMBERSHIP (n1)
```

- **Example 3** – changes the logical server policy for logical server `ls2` to policy `lsp1`.

```
ALTER LOGICAL SERVER ls2 POLICY lsp1
```

Usage

The `SYS.ISYSIQLSMEMBER` system table stores definitions for the logical server memberships.

A member node that is added to or dropped from a logical server starts or stops accepting logical server connections only after the TLV log corresponding to **ALTER LOGICAL SERVER** is played on that node. Existing connections of a logical server continue to run on a node when that node is dropped from the logical server, however, distributed processing is stopped for these connections.

An error is returned if:

- Any `ls-member` specified with the `ADD MEMBERSHIP` clause is already a member of the logical server.
- Any `ls-member` specified with the `DROP MEMBERSHIP` clause is not an existing member of the logical server.
- A logical server membership change causes a node to belong to multiple logical servers assigned to a single login policy. Logical server membership in a login policy cannot overlap.

Permissions

Requires the `MANAGE MULTIPLEX` system privilege.

ALTER LOGIN POLICY Statement

Changes existing login policies or configures logical server access.

Syntax

Syntax 1

```
ALTER LOGIN POLICY policy-name
  { { ADD | DROP | SET } LOGICAL SERVER ls-assignment-list
    [ LOGICAL SERVER ls-override-list ] )
```

```
ls-assignment-list:
  { { ls-name, ... }
    | ALL
    | COORDINATOR
    | SERVER
    | NONE
    | DEFAULT }
```

```
ls-override-list:
  { ls-name, ... }
```

```
ls-name:
  { OPEN | user-defined-ls-name }
```

Syntax 2

```
ALTER LOGIN POLICY policy-name policy-option
```

```
policy-option:
  policy-option-name = policy-option-value
```

```
policy-option-value:
  { UNLIMITED | DEFAULT | value }
```

```
policy-option-name:
  AUTO_UNLOCK_TIME
  | CHANGE_PASSWORD_DUAL_CONTROL
  | DEFAULT_LOGICAL_SERVER
  | LOCKED
  | MAX_CONNECTIONS
  | MAX_DAYS_SINCE_LOGIN
  | MAX_FAILED_LOGIN_ATTEMPTS
  | MAX_NON_DBA_CONNECTIONS
  | PASSWORD_EXPIRY_ON_NEXT_LOGIN
  | PASSWORD_GRACE_TIME
  | PASSWORD_LIFE_TIME
  | ROOT_AUTO_UNLOCK_TIME
  | LDAP_PRIMARY_SERVER
  | LDAP_SECONDARY_SERVER
  | LDAP_AUTO_FAILBACK_PERIOD
```

```
| LDAP_FAILOVER_TO_STD
| LDAP_REFRESH_DN
```

Parameters

- **policy-name** – the name of the login policy. Specify root to modify the root login policy.
- **policy-option-name** – the name of the policy option. See *Login Policy Options* and *LDAP Login Policy Options* for details on each option.
- **policy-option-value** – the value assigned to the login policy option. If you specify UNLIMITED, no limits are used. If you specify DEFAULT, the default limits are used. See *Login Policy Options* and *LDAP Login Policy Options* for supported values for each option.

Applies to

Simplex and multiplex.

Examples

- **Example 1** – see *Logical Server Access Configuration* and *Multiplex Login Policy Configuration* :
- **Example 2** – sets the password_life_time value to UNLIMITED and the max_failed_login_attempts value to 5 in the Test1 login policy:

```
ALTER LOGIN POLICY Test1
password_life_time=UNLIMITED
max_failed_login_attempts=5;
```

Usage

If you do not specify a policy option, values for this login policy are taken from the root login policy. New policies do not inherit the MAX_NON_DBA_CONNECTIONS and ROOT_AUTO_UNLOCK_TIME policy options.

All new databases include a root login policy. You can modify the root login policy values, but you cannot delete the policy.

Permissions

Requires the MANAGE ANY LOGIN POLICY system privilege.

Login Policy Options

Available options for root and user-defined login policies.

Option	Description
AUTO_UNLOCK_TIME	<p>The time period after which locked accounts not granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined in any login policy, including the root login policy.</p> <ul style="list-style-type: none"> • Values – 0 – UNLIMITED • Default – UNLIMITED • Applies to – All users not granted the MANAGE ANY USER system privilege.
CHANGE_PASSWORD_DUAL_CONTROL	<p>Requires input from two users, each granted the CHANGE PASSWORD system privilege, to change the password of another user.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF • Applies to – All users.
DEFAULT_LOGICAL_SERVER	<p>If the connection string specifies no logical server, the user connects to the DEFAULT_LOGICAL_SERVER option specified in the user's login policy.</p> <ul style="list-style-type: none"> • Values – <ul style="list-style-type: none"> • Name of an existing user-defined logical server • ALL – allows access to all logical servers. • AUTO – value of the default logical server in the root login policy. • COORDINATOR – the current coordinator node. • NONE – denies access to any multiplex server. • OPEN – use alone or with the name of a user-defined logical server. Allows access to all multiplex nodes that are not members of any user-defined logical servers. • SERVER – allows access to all of the multiplex nodes, subject to the semantics of the SERVER logical server. • Default – AUTO • Applies to – All users. Requires MANAGE MULTIPLEX system privilege.

Option	Description
LOCKED	<p>If set ON, users cannot establish new connections. This setting temporarily denies access to login policy users. Logical server overrides for this option are not allowed.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF • Applies to – All users except those with the MANAGE ANY USER system privilege.
MAX_CONNECTIONS	<p>The maximum number of concurrent connections allowed for a user. You can specify a per-logical-server setting for this option.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users except those with the SERVER OPERATOR or DROP CONNECTION system privilege.
MAX_DAYS_SINCE_LOGIN	<p>The maximum number of days that can elapse between two successive logins by the same user.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users except those with the MANAGE ANY USER system privilege.
MAX_FAILED_LOGIN_ATTEMPTS	<p>The maximum number of failed attempts, since the last successful attempt, to log into the user account before the account is locked.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users.
MAX_NON_DBA_CONNECTIONS	<p>The maximum number of concurrent connections that a user without SERVER OPERATOR or DROP CONNECTION system privileges can make. This option is supported only in the root login policy.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users except those with the SERVER OPERATOR or DROP CONNECTION privilege.

Option	Description
PASSWORD_EXPIRY_ON_NEXT_LOGIN	<p>If set ON, the user's password expires at the next login.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF • Applies to – All users. <hr/> <p>Note: This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL).</p>
PASSWORD_GRACE_TIME	<p>The number of days before password expiration during which login is allowed but the default post_login procedure issues warnings.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – 0 • Applies to – All users.
PASSWORD_LIFE_TIME	<p>The maximum number of days before a password must be changed.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users.
ROOT_AUTO_UNLOCK_TIME	<p>The time period after which locked accounts granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined only in the root login policy.</p> <ul style="list-style-type: none"> • Values – 0 – UNLIMITED • Default – 15 • Applies to – All users granted the MANAGE ANY USER system privilege.

LDAP Login Policy Options

Available login policy options for LDAP user authentication

Option	Description
LDAP_PRIMARY_SERVER	<p>Specifies the name of the primary LDAP server.</p> <ul style="list-style-type: none"> • Values – n/a • Default – None • Applies to – All users.
LDAP_SECONDARY_SERVER	<p>Specifies the name of the secondary LDAP server.</p> <ul style="list-style-type: none"> • Values – n/a • Default – None • Applies to – All users.
LDAP_AUTO_FAILBACK_PERIOD	<p>Specifies the time period, in minutes, after which automatic failback to the primary server is attempted.</p> <ul style="list-style-type: none"> • Values – 0 - 2147483647 • Default – 15 minutes • Applies to – All users.
LDAP_FAILOVER_TO_STD	<p>Permits authentication with standard authentication when authentication with the LDAP server fails due to system resources, network outage, connection timeouts, or similar system failures. However, it does not permit an actual authentication failure returned from an LDAP server to fail over to standard authentication.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – ON • Applies to – All users.

Option	Description
LDAP_REFRESH_DN	<p>Updates the ldap_refresh_dn value in the ISYSLOGINPOLICYOPTION system table with the current time, stored in Coordinated Universal Time (UTC).</p> <p>Each time a user authenticates with LDAP, if the value of ldap_refresh_dn in ISYSLOGINPOLICYOPTION is more recent than the value of user_dn in ISYSUSER, a search for a new user DN occurs. The user_dn value is then updated with the new user DN and the user_dn_changed_at value is again updated to the current time.</p> <ul style="list-style-type: none"> • Values – NOW • Initial value for ROOT policy – NULL • Initial value for user-defined login policy – Current time stored in UTC • Applies to – All users.

Multiplex Login Policy Configuration

Configure login policies for multiplex servers.

Example

This example overrides the login policy settings on a logical server, increasing the maximum number of connections on logical server ls1:

```
ALTER LOGIN POLICY lp1 max_connections=20 LOGICAL SERVER ls1;
```

Usage

Applies only to multiplex.

Any login management commands you execute on any multiplex server automatically propagate to all servers in the multiplex. For best performance, execute these commands, or any DDL, on the coordinator.

An override at the logical server level override means that a particular login policy option has different settings for different logical servers. SYS.ISYSIQLSLOGINPOLICYOPTION stores login policy option values for logical-server override. For each logical-server override of a login policy option, a corresponding row exists in ISYSIQLSLOGINPOLICYOPTION.

Logical Server Access Configuration

Configure logical server access.

Example 1

Assume that the root login policy allows access to logical servers ls4 and ls5 and login policy lp1 exists with no logical server assignment. The statement below effectively assigns login policy lp1 to logical servers ls4 and ls5.

Assign logical server `ls1` to login policy `lp1`:

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls1
```

Example 2

This statement allows access of logical servers `ls2` and `ls3` from login policy `lp1`:

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls2, ls3
```

Example 3

Modify login policy `lp1` to allow access to `ls3` and `ls4` only:

```
ALTER LOGIN POLICY lp1 ADD LOGICAL SERVER ls4
```

```
ALTER LOGIN POLICY lp1 DROP LOGICAL SERVER ls1, ls2
```

or:

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER ls3, ls4
```

Example 4

Modify login policy `lp1` to deny access to any logical servers:

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER NONE
```

Example 5

Drop current logical server assignments of login policy `lp1` and allow it to inherit the logical server assignments of the root login policy:

```
ALTER LOGIN POLICY lp1 SET LOGICAL SERVER DEFAULT
```

Usage

ADD, **DROP**, or **SET** clauses let you configure the logical server assignments of a login policy:

- **ADD** – adds new logical server assignments to a login policy.
- **DROP** – deletes existing logical server assignments from a login policy.
- **SET** – replaces all logical server assignments for a login policy with a new set of logical server.

Use only one **ADD**, **DROP**, or **SET** clause. Use **SERVER**, **NONE**, and **DEFAULT** clauses only with the **SET** clause. Specify a particular logical server name only once per **ls-assignment list** or **ls-override list**.

An error is returned if:

- Any logical server specified with the **ADD** clause is already assigned to the login policy.
- Any logical server specified with the **DROP** clause is currently not assigned to the login policy.
- Logical server assignment change may cause a membership overlap among assigned logical servers.

`SYS.ISYSIQLOGINPOLICYLSINFO` stores logical server assignment information. For each logical-server override of a login policy option, a corresponding row exists in `ISYSIQLOGINPOLICYLSINFO`.

ALTER LS POLICY Statement

Modifies some or all option values for the root logical server policy or a user-created logical server policy. This statement enforces consistent shared system temporary store settings across physical nodes shared by logical servers.

Syntax

```
ALTER LS POLICY ls-policy-name ls-option-value-list
  [ WITH STOP SERVER ]
```

```
ls-option-value-list:
  { ls-option-name = ls-policy-option-value } ...
```

```
ls-option-name:
  ALLOW_COORDINATOR_AS_MEMBER
  | DQP_ENABLED
  | LOGIN_REDIRECTION
  | REDIRECTION_WAITERS_THRESHOLD
  | TEMP_DATA_IN_SHARED_TEMP
```

Parameters

- **ls-policy-name** – the name of the logical server policy. Specify root to modify the root logical server policy.
- **ls-option-value-list** – the name of the logical server policy option. See *LS Policy Options* for details on each option.
- **ls-policy-option-value** – any unspecified option inherits its value from the root logical server policy. See *LS Policy Options* for supported values for each option
- **WITH STOP SERVER** – automatically shuts down all servers in the logical server when the `TEMP_DATA_IN_SHARED_TEMP` option is changed directly or indirectly.

Applies to

Multiplex only.

Examples

- **Example 1** – alters the logical server policy:

```
ALTER LS POLICY root
ALLOW_COORDINATOR_AS_MEMBER=ON
```

- **Example 2** – alters the logical server policy and causes servers to shut down automatically when the option value changes:

```
ALTER LS POLICY root
TEMP_DATA_IN_SHARED_TEMP=ON WITH STOP SERVER
```

Usage

If you want a smaller IQ_SYSTEM_TEMP dbspace, set TEMP_DATA_IN_SHARED_TEMP to ON, which writes temporary data to IQ_SHARED_TEMP instead of IQ_SYSTEM_TEMP. In a distributed query processing environment, however, setting both DQP_ENABLED and TEMP_DATA_IN_SHARED_TEMP to ON may saturate your SAN with additional data in IQ_SHARED_TEMP, where additional I/O operations against IQ_SHARED_TEMP may adversely affect DQP performance.

Permissions

Requires the MANAGE MULTIPLEX system privilege.

LS Policy Options

Available options for root and user-defined LS policies.

Option	Description
ALLOW_COORDINATOR_AS_MEMBER	<p>Can only be set for the ROOT logical server policy. When ON (the default), the coordinator can be a member of any user-defined logical server. OFF prevents the coordinator from being used as a member of any user-defined logical servers.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – ON
DQP_ENABLED	<p>When set to 0, query processing is not distributed. When set to 1 (the default), query processing is distributed as long as a writable shared temporary file exists. When set to 2, query processing is distributed over the network, and the shared temporary store is not used.</p> <ul style="list-style-type: none"> • Values – 0, 1, 2 • Default – 1

Option	Description
LOGIN_REDIRECTION	<p>When ON, enables login redirection for logical servers governed by specified login policy. When OFF (the default), disables login redirection at the logical server level, allowing external connection management.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF
REDIRECTION_WAITERS_THRESHOLD	<p>Specifies how many connections can queue before SAP Sybase IQ redirects a connection to this logical server to another server. Can be any integer value; default is 5.</p> <ul style="list-style-type: none"> • Values – Integer • Default – 5
TEMP_DATA_IN_SHARED_TEMP	<p>When ON, all temporary table data and eligible scratch data writes to the shared temporary store, provided that the shared temporary store has at least one read-write file added. You must restart all multiplex nodes after setting this option or after adding a read-write file to the shared temporary store. (If the shared temporary store contains no read-write file, or if you do not restart nodes, data is written to IQ_SYSTEM_TEMP instead.) When OFF (the default), all temporary table data and scratch data writes to the local temporary store.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF

ALTER MULTIPLEX RENAME Statement

Renames the multiplex and stores the multiplex name in SYS.ISYSIQINFO system table.

Syntax

```
ALTER MULTIPLEX RENAME multiplex-name
```

Applies to

Multiplex only.

Usage

When a multiplex is created, it is named after the coordinator. This statement is automatically committed.

Permissions

Requires the `MANAGE MULTIPLEX` system privilege.

ALTER MULTIPLEX SERVER Statement

Changes the name, catalog file path, role, or status of the given server.

Syntax

Syntax 1

```
ALTER MULTIPLEX SERVER server-name server-option
```

server-option:

```
{ RENAME new-server-name
| DATABASE 'dbfile'
| ROLE { WRITER | READER | COORDINATOR }
| STATUS { INCLUDED | EXCLUDED }
| ASSIGN AS FAILOVER SERVER
| host-port-list }
```

host-port-list:

```
{ HOST 'hostname' PORT port number ...}
{ PRIVATE HOST 'hostname' PORT port number ...}
```

Syntax 2

```
ALTER MULTIPLEX SERVER server-name PRIVATE NULL
```

Parameters

- **RENAME** – changes the name of the given server. The server automatically shuts down and the next restart requires the new name.
- **DATABASE** – changes the catalog file path for the given server. The server automatically shuts down and the next restart requires the new catalog path. The user must relocate the catalog file.
- **ROLE** – changes the role of the given server. Users cannot change the role of coordinator or role to coordinator. If the role of the writer node changes to reader, the server shuts down.
- **STATUS** – changes the status of the given server. A failover node cannot be excluded unless it is the last node to be excluded. The server automatically shuts down after exclusion. After including a node, you synchronize and restart it.
- **ASSIGN** – designates the given server as the new failover server. The node should not be in the excluded state. The **ASSIGN AS FAILOVER** clause is a standalone clause that cannot be used with any other **ALTER MULTIPLEX SERVER** clause.

The coordinator must be running, but you can run the **ALTER MULTIPLEX SERVER** command from any server in the multiplex. (Run all DDL statements on the coordinator.)

SQL Statements

In all cases except when altering role from reader to writer, the named server is automatically shut down.

- **host-port-list** – Shut down the target server before you exclude it. If you do not, an excluded server automatically shuts down and requires **ALTER MULTIPLEX SERVER *server-name* STATUS INCLUDED** and a synchronize to rejoin the multiplex.

Applies to

Multiplex only.

Examples

- **Example 1** – excludes secondary server `mpx_writer1`:

```
ALTER MULTIPLEX SERVER mpx_writer1 STATUS EXCLUDED
```

Permissions

Requires the **MANAGE MULTIPLEX** system privilege.

ALTER PROCEDURE Statement

Replaces an existing procedure with a modified version. Include the entire modified procedure in the **ALTER PROCEDURE** statement, and reassign user permissions on the procedure.

Syntax

Syntax 1

```
ALTER PROCEDURE [ owner.]procedure-name procedure-definition
```

Syntax 2

```
ALTER PROCEDURE [ owner.]procedure-name  
  REPLICATE { ON | OFF }
```

Syntax 3

```
ALTER PROCEDURE [ owner.]procedure-name  
  SET HIDDEN
```

Syntax 4

```
ALTER PROCEDURE [ owner.]procedure-name  
  RECOMPILE
```

Syntax 5

```
ALTER PROCEDURE  
  [ owner.]procedure-name ( [ parameter, ...] )  
  [ RESULT (result-column, ...)]  
  EXTERNAL NAME 'external-call' [ LANGUAGE JAVA [ environment-name ] ]
```

```
environment-name:
    DISALLOW | ALLOW SERVER SIDE REQUESTS

external-call:
    [column-name:] function-name@library; ...
```

Parameters

- **procedure-definition** – **CREATE PROCEDURE** syntax following the name.
- **REPLICATE** – if a procedure needs to be relocated to other sites using SAP Sybase Replication Server, use the REPLICATE ON clause.
- **SET HIDDEN** – to obfuscate the definition of the associated procedure and cause it to become unreadable. The procedure can be unloaded and reloaded into other databases.

Note: This setting is irreversible. It is recommended that you retain the original procedure definition outside of the database.

- **RECOMPILE** – recompiles a stored procedure. When you recompile a procedure, the definition stored in the catalog is re-parsed and the syntax is verified.
The procedure definition is not changed by recompiling. You can recompile procedures with definitions hidden with the SET HIDDEN clause, but their definitions remain hidden.
- **RESULT** – for procedures that generate a result set but do not include a RESULT clause, the database server attempts to determine the result set characteristics for the procedure and stores the information in the catalog. This can be useful if a table referenced by the procedure has been altered to add, remove, or rename columns since the procedure was created.
- **environment-name** – DISALLOW is the default. ALLOW indicates that server-side connections are allowed.

Note:

- Do not specify ALLOW unless necessary. Use of the ALLOW clause slows down certain types of SAP Sybase IQ table joins.
 - Do not use UDFs with both ALLOW SERVER SIDE REQUESTS and DISALLOW SERVER SIDE REQUESTS clauses in the same query.
-

Usage

The **ALTER PROCEDURE** statement must include the entire new procedure. You can use **PROC** as a synonym for **PROCEDURE**. Both Watcom and Transact-SQL dialect procedures can be altered through the use of **ALTER PROCEDURE**. Existing permissions on the procedure are not changed. If you execute **DROP PROCEDURE** followed by **CREATE PROCEDURE**, execute permissions are reassigned.

SQL Statements

You cannot combine Syntax 2 with Syntax 1.

When using the **ALTER PROCEDURE** statement for table UDFs, the same set of restrictions apply as for the **CREATE PROCEDURE Statement (External Procedures)**.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Alter a Watcom-SQL or Transcat-SQL procedure – Requires one of:

- ALTER ANY PROCEDURE system privilege.
- ALTER ANY OBJECT system privilege.
- You own the procedure.

Alter an external C/C++ or external environment procedure – Requires CREATE EXTERNAL REFERENCE system privilege. Also requires one of:

- ALTER ANY PROCEDURE system privilege.
- ALTER ANY OBJECT system privilege.
- You own the procedure.

See also

- *CREATE PROCEDURE Statement* on page 165

ALTER ROLE Statement

Migrates a compatibility role to a user-defined system role, then automatically drops the compatibility role.

Note: You cannot use the ALTER ROLE statement to migrate SYS_AUTH_SA_ROLE or SYS_AUTH_SSO_ROLE. These roles are automatically migrated when SYS_AUTH_DBA_ROLE is migrated.

Syntax

Syntax 1 – To migrate SYS_AUTH_DBA_ROLE

```
ALTER ROLE predefined_sys_role_name  
MIGRATE TO new_role_name [, new_sa_role_name, new_sso_role_name]
```

Syntax 2 – To migrate all other compatibility roles

```
ALTER ROLE predefined_sys_role_name  
MIGRATE TO new_role_name
```


Parameters

- **predefined_sys_role_name** – the name of a compatibility role that still exists (has not already been dropped) in the database.
- **new_role_name** – the name of the new role cannot begin with the prefix `SYS_` or end with the suffix `_ROLE`.
- **new_sa_role_name** – required only when migrating `SYS_AUTH_DBA_ROLE`. The new role to which the underlying system privileges of `SYS_AUTH_SA_ROLE` are to be migrated to cannot already exist in the database, and the new role name cannot begin with the prefix `SYS_` or end with the suffix `_ROLE`.
- **new_sso_role_name** – required only when migrating `SYS_AUTH_DBA_ROLE`. The new role to which the underlying system privileges of `SYS_AUTH_SSO_ROLE` are to be migrated to cannot already exist in the database, and the new role name cannot begin with the prefix `SYS_` or end with the suffix `_ROLE`.

Examples

- **Example 1** – migrates `SYS_AUTH_DBA_ROLE` to the new roles `Custom_DBA`, `Custom_SA`, and `Custom_SSO` respectively. It then automatically migrates all users, underlying system privileges, and roles granted to `SYS_AUTH_DBA_ROLE` to the applicable new roles. Finally, it drops `SYS_AUTH_DBA_ROLE`, `SYS_AUTH_SA_ROLE`, and `SYS_AUTH_SSO_ROLE`.

```
ALTER ROLE SYS_AUTH_DBA_ROLE
MIGRATE TO Custom_DBA, Custom_SA, Custom_SSO
```

- **Example 2** – migrates `SYS_AUTH_OPERATOR_ROLE` role to the new role `Operator_role`. It then automatically migrates all users, underlying system privileges, and roles granted to `SYS_AUTH_OPERATOR_ROLE` to the new role and drops `SYS_AUTH_OPERATOR_ROLE`.

```
ALTER ROLE SYS_AUTH_OPERATOR_ROLE
MIGRATE TO Operator_role
```

Usage

During the migration process:

- A new user-defined role is created.
- All of the system privileges currently granted to the migrating predefined role are automatically granted to the new user-defined role.
- All users and roles currently granted to the migrating predefined role are automatically granted to the new user-defined role.
- The compatibility role is dropped.

Since no role administrator was specified during the migration process, only global role administrators can manage the new role. Use the CREATE ROLE statement to add role administrators with appropriate administrative rights to the role.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE ROLES system privilege granted with administrative rights.

ALTER SEQUENCE statement

Alters a sequence. This statement applies to SAP Sybase IQ catalog store tables only.

Syntax

```
ALTER SEQUENCE [ owner.] sequence-name
[ RESTART WITH signed-integer ]
[ INCREMENT BY signed-integer ]
[ MINVALUE signed-integer | NO MINVALUE ]
[ MAXVALUE signed-integer | NO MAXVALUE ]
[ CACHE integer | NO CACHE ]
[ CYCLE | NO CYCLE ]
```

Parameters

RESTART WITH clause – Restarts the named sequence with the specified value.

INCREMENT BY clause – Defines the amount the next sequence value is incremented from the last value assigned. The default is 1. Specify a negative value to generate a descending sequence. An error is returned if the INCREMENT BY value is 0.

MINVALUE clause – Defines the smallest value generated by the sequence. The default is 1. An error is returned if MINVALUE is greater than $(2^{63}-1)$ or less than $-(2^{63}-1)$. An error is also returned if MINVALUE is greater than MAXVALUE.

MAXVALUE clause – Defines the largest value generated by the sequence. The default is $2^{63}-1$. An error is returned if MAXVALUE is greater than $2^{63}-1$ or less than $-(2^{63}-1)$.

CACHE clause – Specifies the number of preallocated sequence values that are kept in memory for faster access. When the cache is exhausted, the sequence cache is repopulated and a corresponding entry is written to the transaction log. At checkpoint time, the current value of the cache is forwarded to the ISYSSEQUENCE system table. The default is 100.

CYCLE clause – Specifies whether values should continue to be generated after the maximum or minimum value is reached.

Remarks

If the named sequence cannot be located, an error message is returned.

Privileges

You must be the owner of the sequence, or have one of the following privileges:

```
ALTER ANY SEQUENCE system privilege
ALTER ANY OBJECT system privilege
```

Side effects

None

Standards and compatibility

- **SQL/2008** – The ALTER SEQUENCE statement is part of optional SQL language feature T176 of the SQL/2008 standard. The CACHE clause is a vendor extension.

Example

The following example sets a new maximum value for a sequence named Test:

```
ALTER SEQUENCE Test
    MAXVALUE 1500;
```

ALTER SERVER Statement

Modifies the attributes of a remote server. Changes made by **ALTER SERVER** do not take effect until the next connection to the remote server.

Syntax

```
ALTER SERVER server-name
  [ CLASS 'server-class' ]
  [ USING 'connection-info' ]
  [ CAPABILITY 'cap-name' { ON | OFF } ]
  [ CONNECTION CLOSE [ CURRENT | ALL | connection-id ] ]

server-class:
  { ASAJDBC
    | ASEJDBC
    | SAODBC
    | ASEODBC
    | DB2ODBC
    | MSSODBC
    | ORAODBC
    | ODBC }

connection-info:
  { machine-name:port-number [ /dbname ] | data-source-name }
```

Parameters

- **cap-name** – the name of a server capability
- **CLASS** – changes the server class. For more information on server classes.
- **USING** – if a JDBC-based server class is used, the USING clause is *hostname:port-number [/dbname]* where:
 - **hostname** – the machine on which the remote server runs.
 - **portnumber** – the TCP/IP port number on which the remote server listens. The default port number for SAP Sybase IQ and SQL Anywhere is 2638.
 - **dbname** – for SQL Anywhere remote servers, if you do not specify a *dbname*, the default database is used. For Adaptive Server Enterprise, the default is the master database, and an alternative to using *dbname* is to another database by some other means (for example, in the **FORWARD TO** statement).

If an ODBC-based server class is used, the USING clause is the *data-source-name*, which is the ODBC Data Source Name.

- **CAPABILITY** – turns a server capability ON or OFF. Server capabilities are stored in the system table SYSCAPABILITY. The names of these capabilities are stored in the system table SYSCAPABILITYNAME. The SYSCAPABILITY table contains no entries for a remote server until the first connection is made to that server. At the first connection, SAP Sybase IQ interrogates the server about its capabilities and then populates SYSCAPABILITY. For subsequent connections, the server's capabilities are obtained from this table.

In general, you need not alter a server's capabilities. It might be necessary to alter capabilities of a generic server of class ODBC.

- **CONNECTION CLOSE** – when a user creates a connection to a remote server, the remote connection is not closed until the user disconnects from the local database. The CONNECTION CLOSE clause allows you to explicitly close connections to a remote server. You may find this useful when a remote connection becomes inactive or is no longer needed.

These SQL statements are equivalent and close the current connection to the remote server:

```
ALTER SERVER server-name CONNECTION CLOSE
ALTER SERVER server-name CONNECTION CLOSE CURRENT
```

You can close both ODBC and JDBC connections to a remote server using this syntax. You do not need the SERVER OPERATOR system privilege to execute either of these statements.

You can also disconnect a specific remote ODBC connection by specifying a connection ID, or disconnect all remote ODBC connections by specifying the ALL keyword. If you attempt to close a JDBC connection by specifying the connection ID or the ALL keyword,

an error occurs. When the connection identified by *connection-id* is not the current local connection, the user must have the SERVER OPERATOR system privilege to be able to close the connection.

Examples

- **Example 1** – changes the server class of the Adaptive Server Enterprise server named `ase_prod` so its connection to SAP Sybase IQ is ODBC-based. The Data Source Name is `ase_prod`.

```
ALTER SERVER ase_prod
CLASS 'ASEODBC'
USING 'ase_prod'
```

- **Example 2** – changes a capability of server `infodc`:

```
ALTER SERVER infodc
CAPABILITY 'insert select' OFF
```

- **Example 3** – closes all connections to the remote server named `rem_test`:

```
ALTER SERVER rem_test
CONNECTION CLOSE ALL
```

- **Example 4** – closes the connection to the remote server named `rem_test` that has the connection ID 142536:

```
ALTER SERVER rem_test
CONNECTION CLOSE 142536
```

Usage

Side effects:

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Open Client/Open Server.

Permissions

Requires the SERVER OPERATOR system privilege.

See also

- *CREATE SERVER Statement* on page 192
- *DROP SERVER Statement* on page 269

ALTER SERVICE Statement

Causes the database server to act as a Web server

Syntax

```
ALTER SERVICE service-name
  [ TYPE service-type-string ]
  [ attributes ]
  [ AS statement' ]

attributes:
  [ AUTHORIZATION { ON | OFF } ]
  [ SECURE { ON | OFF } ]
  [ USER { user-name | NULL } ]
  [ URL [ PATH/ ] { ON | OFF | ELEMENTS } ]
  [ USING { SOAP-prefix | NULL } ]

service-type-string:
  { 'RAW '
  | 'HTML '
  | 'XML '
  | 'SOAP '
  | 'DISH ' }
```

Parameters

- **service-type-string** – Web service names may be any sequence of alphanumeric characters or “/”, “-”, “_”, “:”, “!”, “~”, “*”, “”, “(”, or “”)”, except that the first character cannot begin with a slash (/) and the name cannot contain two or more consecutive slash characters.
- **AUTHORIZATION** – determines whether users must specify a user name and password when connecting to the service. The default value is ON.
 - If authorization is OFF, the AS clause is required and a single user must be identified by the USER clause. All requests are run using that user’s account and permissions.
 - If authorization is ON, all users must provide a user name and password. Optionally, you can limit the users that are permitted to use the service by providing a user or role name using the USER clause. If the user name is NULL, all known users can access the service.

Run production systems with authorization turned on. Grant permission to use the service by adding users to a role.

- **SECURE** – indicates whether unsecure connections are accepted. ON indicates that only HTTPS connections are to be accepted. Service requests received on the HTTP port are

automatically redirected to the HTTPS port. If set to OFF, both HTTP and HTTPS connections are accepted. The default value is OFF.

- **USER** – if authorization is disabled, this parameter becomes mandatory and specifies the user ID used to execute all service requests. If authorization is enabled (the default), this optional clause identifies the user or role permitted access to the service. The default value is NULL, which grants access to all users.
- **URL** – determines whether URI paths are accepted and, if so, how they are processed. OFF indicates that nothing must follow the service name in a URI request. ON indicates that the remainder of the URI is interpreted as the value of a variable named *url*. ELEMENTS indicates that the remainder of the URI path is to be split at the slash characters into a list of up to 10 elements. The values are assigned to variables named *url* plus a numeric suffix of between 1 and 10; for example, the first three variable names are *url1*, *url2*, and *url3*. If fewer than 10 values are supplied, the remaining variables are set to NULL. If the service name ends with the character /, then URL must be set to OFF. The default value is OFF.
- **USING** – applies only to DISH services. The parameter specifies a name prefix. Only SOAP services whose names begin with this prefix are handled.
- **service-type-string** – identifies the type of the service. The type must be one of the listed service types. There is no default value.
 - **RAW** – sends the result set to the client without any further formatting. You can produce formatted documents by generating the required tags explicitly within your procedure.
 - **HTML** – formats the result set of a statement or procedure into an HTML document that contains a table.
 - **XML** – assumes the result set is an XML format. If it is not already so, it is automatically converted to XML RAW format.
 - **SOAP** – formats the result set as a Simple Object Access Protocol (SOAP) response. The request must be a valid SOAP request. For more information about the SOAP standards, see www.w3.org/TR/SOAP
 - **DISH** – determine SOAP Handler, or DISH, service acts as a proxy for one or more SOAP services. In use, it acts as a container that holds and provides access to a number of SOAP services. A Web Services Description Language (WSDL) file is automatically generated for each of the included SOAP services. The included SOAP services are identified by a common prefix, which must be specified in the **USING** clause.
- **statement** – if the statement is NULL, the URI must specify the statement to be executed. Otherwise, the specified SQL statement is the only one that can be executed through the service. The statement is mandatory for SOAP services, and ignored for DISH services. The default value is NULL.

All services that are run in production systems must define a statement. The statement can be NULL only if authorization is enabled.

Examples

- **Example 1** – to set up a Web server quickly, starts a database server with the **-xs** switch, then execute these statements:

```
CREATE SERVICE tables TYPE 'HTML'  
ALTER SERVICE tables  
AUTHORIZATION OFF  
USER DBA  
AS SELECT * FROM SYS.ISYSTABAfter executing these statements, use  
any Web browser to open the URL http://localhost/tables.
```

Usage

You cannot rename Web services.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the MANAGE ANY WEB SERVICE system privilege.

See also

- *CREATE SERVICE Statement* on page 194
- *DROP SERVICE Statement* on page 269

ALTER SPATIAL REFERENCE SYSTEM Statement

Changes the settings of an existing spatial reference system. See the Remarks section for considerations before altering a spatial reference system

Syntax

ALTER SPATIAL REFERENCE SYSTEM

```
    srs-name  
    [ srs-attribute [ srs-attribute ... ] ]  
  
srs-attribute:  
    SRID srs-id  
    | DEFINITION { definition-string | NULL }  
    | ORGANIZATION { organization-name IDENTIFIED BY organization-srs-id  
    | NULL }  
    | TRANSFORM DEFINITION { transform-definition-string | NULL }  
    | LINEAR UNIT OF MEASURE linear-unit-name  
    | ANGULAR UNIT OF MEASURE { angular-unit-name | NULL }  
    | TYPE { ROUND EARTH | PLANAR }  
    | COORDINATE coordinate-name { UNBOUNDED | BETWEEN low-number
```



```

AND high-number }
| ELLIPSOID SEMI MAJOR AXIS semi-major-axis-length { SEMI MINOR AXIS
semi-minor-axis-length
| INVERSE FLATTENING inverse-flattening-ratio }
| SNAP TO GRID { grid-size | DEFAULT }
| TOLERANCE { tolerance-distance | DEFAULT }
| POLYGON FORMAT polygon-format
| STORAGE FORMAT storage-format

grid-size:
  DOUBLE : usually between 0 and 1

axis-order:
  { 'x/y/z/m' | 'long/lat/z/m' | 'lat/long/z/m' }

polygon-format:
  { 'CounterClockWise' | 'Clockwise' | 'EvenOdd' }

storage-format:
  { 'Internal' | 'Original' | 'Mixed' }

```

Parameters

- **IDENTIFIED BY** – the SRID number for the spatial reference system.
- **DEFINITION** – set, or override, default coordinate system settings. If any attribute is set in a clause other than the DEFINITION clause, it takes the value specified in the other clause regardless of what is specified in the DEFINITION clause.

definition-string is a string in the Spatial Reference System Well Known Text syntax as defined by SQL/MM and OGC. For example, the following query returns the definition for WGS 84.

```

SELECT ST_SpatialRefSys::ST_FormatWKT( definition )
FROM ST_SPATIAL_REFERENCE_SYSTEMS
WHERE srs_id=4326;

```

In Interactive SQL, if you double-click the value returned, an easier to read version of the value appears.

When the DEFINITION clause is specified, *definition-string* is parsed and used to choose default values for attributes. For example, *definition-string* may contain an AUTHORITY element that defines the organization-name and *organization-srs-id*.

Parameter values in *definition-string* are overridden by values explicitly set using the SQL statement clauses. For example, if the ORGANIZATION clause is specified, it overrides the value for ORGANIZATION in *definition-string*.

- **ORGANIZATION** – information about the organization that created the spatial reference system that the spatial reference system is based on.
- **TRANSFORM DEFINITION** – a description of the transform to use for the spatial reference system. Currently, only the PROJ.4 transform is supported. The transform definition is used by the ST_Transform method when transforming data between spatial

reference systems. Some transforms may still be possible even if there is no transform-definition-string defined.

- **LINEAR UNIT OF MEASURE** – the linear unit of measure for the spatial reference system. The value you specify must match a linear unit of measure defined in the `ST_UNITS_OF_MEASURE` system view.

If this clause is not specified, and is not defined in the `DEFINITION` clause, the default is `METRE`. To add predefined units of measure to the database, use the `sa_install_feature` system procedure.

To add custom units of measure to the database, use the `CREATE SPATIAL UNIT OF MEASURE` statement.

Note: While both `METRE` and `METER` are accepted spellings, `METRE` is preferred as it conforms to the `SQL/MM` standard.

- **ANGULAR UNIT OF MEASURE** – the angular unit of measure for the spatial reference system. The value you specify must match an angular unit of measure defined in the `ST_UNITS_OF_MEASURE` system table.

If this clause is not specified, and is not defined in the `DEFINITION` clause, the default is `DEGREE` for geographic spatial reference systems and `NULL` for non-geographic spatial reference systems.

The angular unit of measure must be non-`NULL` for geographic spatial reference systems and it must be `NULL` for non-geographic spatial reference systems.

The angular unit of measure must be non-`NULL` for geographic spatial reference systems and it must be `NULL` for non-geographic spatial reference systems. To add predefined units of measure to the database, use the `sa_install_feature` system procedure.

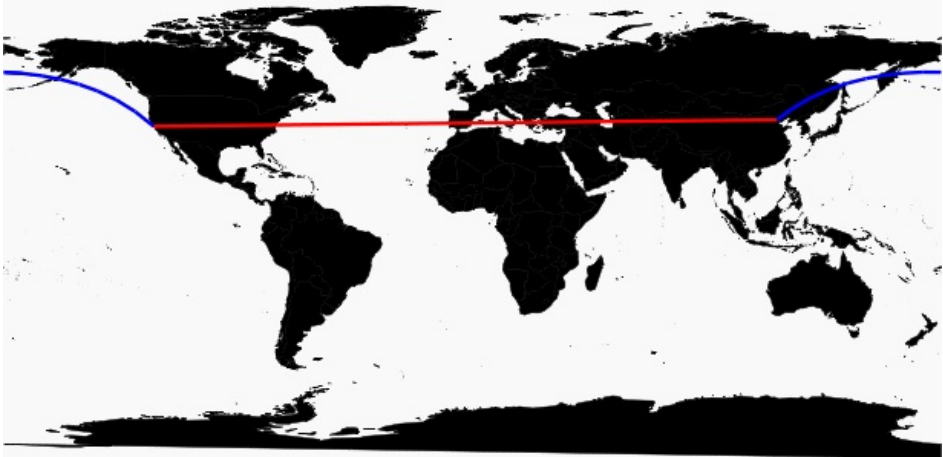
To add custom units of measure to the database, use the `CREATE SPATIAL UNIT OF MEASURE` statement.

- **TYPE** – control how the SRS interprets lines between points. For geographic spatial reference systems, the `TYPE` clause can specify either `ROUND EARTH` (the default) or `PLANAR`. The `ROUND EARTH` model interprets lines between points as great elliptic arcs. Given two points on the surface of the Earth, a plane is selected that intersects the two points and the center of the Earth. This plane intersects the Earth, and the line between the two points is the shortest distance along this intersection.

For two points that lie directly opposite each other, there is not a single unique plane that intersects the two points and the center of the Earth. Line segments connecting these antipodal points are not valid and give an error in the `ROUND EARTH` model.

The `ROUND EARTH` model treats the Earth as a spheroid and selects lines that follow the curvature of the Earth. In some cases, it may be necessary to use a planar model where a line between two points is interpreted as a straight line in the equirectangular projection where `x=long`, `y=lat`.

In the following example, the blue line shows the line interpretation used in the ROUND EARTH model and the red line shows the corresponding PLANAR model.



The PLANAR model may be used to match the interpretation used by other products. The PLANAR model may also be useful because there are some limitations for methods that are not supported in the ROUND EARTH model (such as `ST_Area`, `ST_ConvexHull`) and some are partially supported (`ST_Distance` only supported between point geometries). Geometries based on circularstrings are not supported in ROUND EARTH spatial reference systems.

For non-geographic SRSSs, the type must be PLANAR (and that is the default if the TYPE clause is not specified and either the DEFINITION clause is not specified or it uses a non-geographic definition).

- **COORDINATE** – the bounds on the spatial reference system's dimensions. coordinate-name is the name of the coordinate system used by the spatial reference system. For non-geographic coordinate systems, coordinate-name can be x, y, or m. For geographic coordinate systems, coordinate-name can be LATITUDE, LONGITUDE, z, or m.

Specify UNBOUNDED to place no bounds on the dimensions. Use the BETWEEN clause to set low and high bounds.

The X and Y coordinates must have associated bounds. For geographic spatial reference systems, the longitude coordinate is bounded between -180 and 180 degrees and the latitude coordinate is bounded between -90 and 90 degrees by default unless COORDINATE clause overrides these settings. For non-geographic spatial reference systems, the CREATE statement must specify bounds for both X and Y coordinates.

LATITUDE and LONGITUDE are used for geographic coordinate systems. The bounds for LATITUDE and LONGITUDE default to the entire Earth, if not specified.

- **ELLIPSOID** – the values to use for representing the Earth as an ellipsoid for spatial reference systems of type ROUND EARTH. If the DEFINITION clause is present, it can

specify ellipsoid definition. If the **ELLIPSOID** clause is specified, it overrides this default ellipsoid.

The Earth is not a perfect sphere because the rotation of the Earth causes a flattening so that the distance from the center of the Earth to the North or South pole is less than the distance from the center to the equator. For this reason, the Earth is modeled as an ellipsoid with different values for the semi-major axis (distance from center to equator) and semi-minor axis (distance from center to the pole). It is most common to define an ellipsoid using the semi-major axis and the inverse flattening, but it can instead be specified using the semi-minor axis (for example, this approach must be used when a perfect sphere is used to approximate the Earth). The semi-major and semi-minor axes are defined in the linear units of the spatial reference system, and the inverse flattening (1/f) is a ratio:

$$1/f = (\text{semi-major-axis}) / (\text{semi-major-axis} - \text{semi-minor-axis})$$

product-name uses the ellipsoid definition when computing distance in geographic spatial reference systems.

- **SNAP TO GRID** – flat-Earth (planar) spatial reference systems, use the **SNAP TO GRID** clause to define the size of the grid SAP Sybase IQ uses when performing calculations. By default, SAP Sybase IQ selects a grid size so that 12 significant digits can be stored at all points in the space bounds for X and Y. For example, if a spatial reference system bounds X between -180 and 180 and Y between -90 and 90, then a grid size of 0.000000001 (1E-9) is selected.
- **TOLERANCE** – flat-Earth (planar) spatial reference systems, use the **TOLERANCE** clause to specify the precision to use when comparing points. If the distance between two points is less than tolerance-distance, the two points are considered equal. Setting tolerance-distance allows you to control the tolerance for imprecision in the input data or limited internal precision. By default, tolerance-distance is set to be equal to grid-size.

When set to 0, two points must be exactly equal to be considered equal.

For round-Earth spatial reference systems, **TOLERANCE** must be set to 0.

- **POLYGON FORMAT** – internally, SAP Sybase IQ interprets polygons by looking at the orientation of the constituent rings. As one travels a ring in the order of the defined points, the inside of the polygon is on the left side of the ring. The same rules are applied in **PLANAR** and **ROUND EARTH** spatial reference systems.

The interpretation used by SAP Sybase IQ is a common but not universal interpretation. Some products use the exact opposite orientation, and some products do not rely on ring orientation to interpret polygons. The **POLYGON FORMAT** clause can be used to select a polygon interpretation that matches the input data, as needed. The following values are supported:

- **CounterClockwise** – input follows SAP Sybase IQ's internal interpretation: the inside of the polygon is on the left side while following ring orientation.

- **Clockwise** – input follows the opposite of SAP Sybase IQ's approach: the inside of the polygon is on the right side while following ring orientation.
- **EvenOdd** – (default) The orientation of rings is ignored and the inside of the polygon is instead determined by looking at the nesting of the rings, with the exterior ring being the largest ring and interior rings being smaller rings inside this ring. A ray is traced from a point within the rings and radiating outward crossing all rings. If the number the ring being crossed is an even number, it is an outer ring. If it is odd, it is an inner ring.
- **STORAGE FORMAT** – control what is stored when spatial data is loaded into the database. Possible values are:
 - **Internal** – SAP Sybase IQ stores only the normalized representation. Specify this when the original input characteristics do not need to be reproduced. This is the default for planar spatial reference systems (TYPE PLANAR).
 - **Original** – SAP Sybase IQ stores only the original representation. The original input characteristics can be reproduced, but all operations on the stored values must repeat normalization steps, possibly slowing down operations on the data.
 - **Mixed** – SAP Sybase IQ stores the internal version and, if it is different from the original version, SQL Anywhere stores the original version as well. By storing both versions, the original representation characteristics can be reproduced and operations on stored values do not need to repeat normalization steps. However, storage requirements may increase significantly because potentially two representations are being stored for each geometry. Mixed is the default format for round-Earth spatial reference systems (TYPE ROUND EARTH).

Examples

- **Example** – changes the polygon format of a fictitious spatial reference system named mySpatialRef to EvenOdd:

```
ALTER SPATIAL REFERENCE SYSTEM mySpatialRef
POLYGON FORMAT 'EvenOdd';
```

Usage

You cannot alter a spatial reference system if there is existing data that references it. For example, if you have a column declared as ST_Point(SRID=8743), you cannot alter the spatial reference system with SRID 8743. This is because many spatial reference system attributes, such as storage format, impact the storage format of the data. If you have data that references the SRID, create a new spatial reference system and transform the data to the new SRID.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- You are the owner of the spatial reference system.
- ALTER privilege on the spatial reference system
- MANAGE ANY SPATIAL OBJECT system privilege
- ALTER ANY OBJECT system privilege.

ALTER TABLE Statement

Modifies a table definition.

Syntax

Syntax 1

```
ALTER TABLE table_name ALTER OWNER TO new_owner
  [ { PRESERVE | DROP } PERMISSIONS ]
  [ { PRESERVE | DROP } FOREIGN KEYS ]
```

Syntax 2

```
ALTER TABLE [ owner. ] table-name
  | { ENABLE | DISABLE } RLV STORE
  { alter-clause, ... }
```

alter-clause:

ADD create-clause

```
| ALTER column-name column-alteration
| ALTER [ CONSTRAINT constraint-name ] CHECK ( condition )
| DROP drop-object
| RENAME rename-object
| move-clause
| SPLIT PARTITION range-partition-name
  INTO ( range-partition-decl-1, range-partition-decl-2 )
| MERGE PARTITION partition-name-1 INTO partition-name-2
| UNPARTITION
| PARTITION BY
  range-partitioning-scheme
```

create-clause:

```
column-name column-definition [ column-constraint ]
| table-constraint
| [ PARTITION BY ] range-partitioning-scheme
```

column definition:

```
column-name data-type [ NOT NULL | NULL ]
[ IN dbspace-name ]
[ DEFAULT default-value | IDENTITY ]
```

column-constraint:

```
[ CONSTRAINT constraint-name ]
{ UNIQUE
  | PRIMARY KEY
  | REFERENCES table-name [ ( column-name ) ] [ actions ]
  | CHECK ( condition )
```

```

    | IQ UNIQUE ( integer )
  }

table-constraint:
  [ CONSTRAINT constraint-name ]
  { UNIQUE ( column-name [ , ... ] )
    | PRIMARY KEY ( column-name [ , ... ] )
    | foreign-key-constraint
    | CHECK ( condition )
  }

foreign-key-constraint:
  FOREIGN KEY [ role-name ] [ ( column-name [ , ... ] ) ]
  ... REFERENCES table-name [ ( column-name [ , ... ] ) ]
  ... [ actions ]

actions:
  [ ON { UPDATE | DELETE } { RESTRICT } ]

column-alteration:
  { column-data-type | alterable-column-attribute } [ alterable-column-attribute ... ]

  | ADD [ constraint-name ] CHECK ( condition )
  | DROP { DEFAULT | CHECK | CONSTRAINT constraint-name }

alterable-column-attribute:
  [ NOT ] NULL
  | DEFAULT default-value
  | [ CONSTRAINT constraint-name ] CHECK { NULL | ( condition )
  }

default-value:
  CURRENT { DATABASE | DATE | REMOTE USER | TIME | TIMESTAMP | USER |
PUBLISHER )
  | string
  | global variable
  | [ - ] number
  | ( constant-expression )
  | built-in-function ( constant-expression )
  | AUTOINCREMENT
  | NULL
  | TIMESTAMP
  | LAST USER
  | USER

drop-object:
  { column-name
  | CHECK constraint-name
  | CONSTRAINT
  | UNIQUE ( index-columns-list )
  | PRIMARY KEY
  | FOREIGN KEY fkey-name
  | [ PARTITION ] range-partition-name
  }

rename-object:

```

```

new-table-name
| column-name TO new-column-name
| CONSTRAINT constraint-name TO new-constraint-name
| [ PARTITION ] range-partition-name TO new-range-partition-name

move-clause:
{ ALTER column-name
  MOVE
  { PARTITION ( range-partition-name TO new-dbspace-name)
    | TO new-dbspace-name }
  }
| MOVE PARTITION range-partition-name TO new-dbspace-name
| MOVE TO new-dbspace-name
| MOVE METADATA TO new-dbspace-name
}

range-partitioning-scheme:
RANGE( partition-key )
( range-partition-decl [, range-partition-decl ... ] )

partition-key:
column-name

range-partition-decl:
range-partition-name VALUES <= ( {constant | MAX } ) [ IN dbspace-
name ]

```

Parameters

- **{ ENABLE | DISABLE } RLV STORE** – registers this table with the RLV store for real-time in-memory updates. Not supported for IQ temporary tables. This value overrides the value of the database option **BASE_TABLES_IN_RLV**. Requires the CREATE TABLE system privilege and CREATE permissions on the RLV store dbspace to set this value to ENABLE.
- **ADD column-definition [column-constraint]** – add a new column to the table.

The table must be empty to specify NOT NULL. The table might contain data when you add an IDENTITY or DEFAULT AUTOINCREMENT column. If the column has a default IDENTITY value, all rows of the new column are populated with sequential values. You can also add FOREIGN constraint as a column constraint for a single column key. The value of the IDENTITY/DEFAULT AUTOINCREMENT column uniquely identifies every row in a table.

The IDENTITY/DEFAULT AUTOINCREMENT column stores sequential numbers that are automatically generated during inserts and updates. DEFAULT AUTOINCREMENT columns are also known as IDENTITY columns. When using IDENTITY/DEFAULT AUTOINCREMENT, the column must be one of the integer data types, or an exact numeric type, with scale 0. See *CREATE TABLE Statement* for more about column constraints and IDENTITY/DEFAULT AUTOINCREMENT columns.

IQ UNIQUE constraint – Defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit FP. An IQ UNIQUE(*n*) value explicitly set to

0 loads the column as Flat FP. Columns without an IQ UNIQUE constraint implicitly load as NBit up to the limits defined by the FP_NBIT_AUTOSIZE_LIMIT, FP_NBIT_LOOKUP_MB, and FP_NBIT_ROLLOVER_MAX_MB options.

Using IQ UNIQUE with an *n* value less than the FP_NBIT_AUTOSIZE_LIMIT is not necessary. Auto-size functionality automatically sizes all low or medium cardinality columns as NBit. Use IQ UNIQUE in cases where you want to load the column as Flat FP or when you want to load a column as NBit when the number of distinct values exceeds the FP_NBIT_AUTOSIZE_LIMIT.

Note:

- Consider memory usage when specifying high IQ UNIQUE values. If machine resources are limited, avoid loads with FP_NBIT_ENFORCE_LIMITS='OFF' (default).
Prior to SAP Sybase IQ 16.0, an IQ UNIQUE *n* value > 16777216 would rollover to Flat FP. In 16.0, larger IQ UNIQUE values are supported for tokenization, but may require significant memory resource requirements depending on cardinality and column width.
- BIT, BLOB, and CLOB data types do not support NBit dictionary compression. If FP_NBIT_IQ15_COMPATIBILITY='OFF', a non-zero IQ UNIQUE column specification in a CREATE TABLE or ALTER TABLE statement that includes these data types returns an error.

-
- **ALTER *column-name* column-alteration** – change the column definition:
 - **SET DEFAULT *default-value*** – Change the default value of an existing column in a table. You can also use the MODIFY clause for this task, but ALTER is ISO/ANSI SQL compliant, and MODIFY is not. Modifying a default value does not change any existing values in the table.
 - **DROP DEFAULT** – Remove the default value of an existing column in a table. You can also use the MODIFY clause for this task, but ALTER is ISO/ANSI SQL compliant, and MODIFY is not. Dropping a default does not change any existing values in the table.
 - **ADD** – Add a named constraint or a CHECK condition to the column. The new constraint or condition applies only to operations on the table after its definition. The existing values in the table are not validated to confirm that they satisfy the new constraint or condition.
 - **CONSTRAINT *column-constraint-name*** – The optional column constraint name lets you modify or drop individual constraints at a later time, rather than having to modify the entire column constraint.
 - **[CONSTRAINT *constraint-name*] CHECK (*condition*)** – Use this clause to add a CHECK constraint on the column.

- **SET COMPUTE** (*expression*) – Change the expression associated with a computed column. The values in the column are recalculated when the statement is executed, and the statement fails if the new expression is invalid.
- **DROP COMPUTE** – Change a column from being a computed column to being a non-computed column. This statement does not change any existing values in the table.
- **ADD table-constraint** – add a constraint to the table.

You can also add a foreign key constraint as a table constraint for a single-column or multicolumn key. If PRIMARY KEY is specified, the table must not already have a primary key created by the CREATE TABLE statement or another ALTER TABLE statement. See *CREATE TABLE Statement* for a full explanation of table constraints.

Note: You cannot MODIFY a table or column constraint. To change a constraint, DELETE the old constraint and ADD the new constraint.

- **DROP drop-object** – drops a table object:
 - **DROP column-name** – Drop the column from the table. If the column is contained in any multicolumn index, uniqueness constraint, foreign key, or primary key, then the index, constraint, or key must be deleted before the column can be deleted. This does not delete CHECK constraints that refer to the column. An IDENTITY/DEFAULT AUTOINCREMENT column can only be deleted if IDENTITY_INSERT is turned off and the table is not a local temporary table.
 - **DROP CHECK** – Drop all check constraints for the table. This includes both table check constraints and column check constraints.
 - **DROP CONSTRAINT constraint-name** – Drop the named constraint for the table or specified column.
 - **DROP UNIQUE (column-name, ...)** – Drop the unique constraints on the specified column(s). Any foreign keys referencing the unique constraint (rather than the primary key) are also deleted. Reports an error if there are associated foreign-key constraints. Use ALTER TABLE to delete all foreign keys that reference the primary key before you delete the primary key constraint.
 - **DROP PRIMARY KEY** – Drop the primary key. All foreign keys referencing the primary key for this table are also deleted. Reports an error if there are associated foreign key constraints. If the primary key is unenforced, DELETE returns an error if associated unenforced foreign key constraints exist.
 - **DROP FOREIGN KEY role-name** – Drop the foreign key constraint for this table with the given role name. Retains the implicitly created non-unique HG index for the foreign key constraint. Users can explicitly remove the HG index with the DROP INDEX statement.
 - **DROP [PARTITION]** – Drop the specified partition. The rows in partition P1 are deleted and the partition definition is dropped. You cannot drop the last partition because dropping the last partition would transform a partitioned table to a non-partitioned table. (To merge a partitioned table, use an UNPARTITION clause instead.) For example:

```
CREATE TABLE foo (c1 INT, c2 INT)
PARTITION BY RANGE (c1)
(P1 VALUES <= (100) IN dbbsp1,
 P2 VALUES <= (200) IN dbbsp2,
 P3 VALUES <= (MAX) IN dbbsp3
 ) IN dbbsp4);
LOAD TABLE ...
ALTER TABLE DROP PARTITION P1;
```

- **RENAME *rename-object*** – renames an object in the table:
 - **RENAME *new-table-name*** – Change the name of the table to the *new-table-name*. Any applications using the old table name must be modified. Also, any foreign keys that were automatically assigned the same name as the old table name do not change names.
 - **RENAME *column-name* TO *new-column-name*** – Change the name of the column to *new-column-name*. Any applications using the old column name must be modified.
 - **RENAME [PARTITION]** – Rename an existing partition.
 - **RENAME *constraint-name* TO *new-constraint-name*** – Change the name of the constraint to *new-constraint-name*. Any applications using the old constraint name must be modified.
- **MOVE clause** – moves a table object. A table object can only reside in one dbspace. Any type of ALTER MOVE blocks any modification to the table for the entire duration of the move.
 - **MOVE TO** – Move all table objects including columns, indexes, unique constraints, primary key, foreign keys, and metadata resided in the same dbspace as the table is mapped to the new dbspace. The ALTER Column MOVE TO clause cannot be requested on a partitioned table.
 - **MOVE TABLE METADATA** – Move the metadata of the table to a new dbspace. For a partitioned table, MOVE TABLE METADATA also moves metadata that is shared among partitions.
 - **MOVE PARTITION** – Move the specified partition to the new dbspace.
- **PARTITION BY RANGE** – maps data to partitions based on a range of partition keys established for each partition.

A non-partitioned table can be partitioned if all existing rows belong to the first partition. You can specify a different dbspace for the first partition than the dbspace of the column or table. But existing rows are not moved. Instead, the proper dbspace for the column/partition is kept in `SYS.ISYSIQPARTITIONCOLUMN` for existing columns. Only the default or max identity column(s) that are added later for the first partition are stored in the specified dbspace for the first partition.

Note: ALTER TABLE does not support hash partitioning, hash-range partitioning, or sub-partitioning.

- **MERGE PARTITION** – merge *partition-name-1* into *partition-name-2*. Two partitions can be merged if they are adjacent partitions and the data resides on the same dbspace. You can only merge a partition with a lower partition value into the adjacent partition with a higher partition value. Note that the server does not check CREATE permission on the dbspace into which the partition is merged. For an example of how to create adjacent partitions, see CREATE TABLE Statement examples.
- **RENAME PARTITION** – rename an existing PARTITION.
- **UNPARTITION** – remove partitions from a partitioned table. Each column is placed in a single dbspace. Note that the server does not check CREATE permission on the dbspace to which data of all partitions is moved. ALTER TABLE UNPARTITION blocks all database activities.
- **ALTER OWNER** – change the owner of a table. The **ALTER OWNER** clause may not be used in conjunction with any other [alter-clause] clauses of the ALTER TABLE statement.
 - [**PRESERVE | DROP**] **PERMISSIONS** – If you do not want the new owner to have the same privileges as the old owner, use the DROP permissions clause (default) to drop all explicitly-granted privileges that allow a user access to the table. Implicitly-granted privileges given to the owner of the table are given to the new owner and dropped from the old owner.
 - [**PRESERVE | DROP**] **FOREIGN KEYS** – If you want to prevent the new owner from accessing data in referenced tables, use the DROP FOREIGN KEYS clause (default) to drop all foreign keys within the table, as well as all foreign keys referring to the table. Use of the PRESERVE FOREIGN KEYS clause with the DROP PERMISSIONS clause fails unless all referencing tables are owned by the new owner.

The **ALTER TABLE ALTER OWNER** statement fails if:

- Another table with the same name as the original table exists and is owned by the new user.
- The PRESERVE FOREIGN KEYS and PRESERVE PERMISSIONS clauses are both specified and there is a foreign key owned by a user other than the new table owner referencing the table that relies on implicitly-granted permissions (such as those given to the owner of a table). To avoid this failure, explicitly grant SELECT permissions to the referring table's original owner, or drop the foreign keys.
- The PRESERVE FOREIGN KEYS clause is specified, but the PRESERVE PERMISSIONS clause is NOT, and there is a foreign key owned by a user other than the new table owner referencing the table. To avoid this failure, drop the foreign keys.
- The PRESERVE FOREIGN KEYS clause is specified and the table contains a foreign key that relies on implicitly-granted permissions (such as those given to the owner of a table). To avoid this failure, explicitly GRANT SELECT permissions to the new owner on the referenced table, or drop the foreign keys.
- The table contains a column with a default value that refers to a sequence, and the USAGE permission of the sequence generator relies on implicitly-granted permissions

(such as those given to the owner of a sequence). To avoid this failure, explicitly grant USAGE permission on the sequence generator to the new owner of the table.

- Enabled materialized views that depend on the original table exist.

Examples

- **Example 1** – adds a new column to the `Employees` table showing which office they work in:

```
ALTER TABLE Employees
ADD office CHAR(20)
```

- **Example 2** – drops the `office` column from the `Employees` table:

```
ALTER TABLE Employees
DROP office
```

- **Example 3** – Adds a column to the `Customers` table assigning each customer a sales contact:

```
ALTER TABLE Customers
ADD SalesContact INTEGER
REFERENCES Employees (EmployeeID)
```

- **Example 4** – adds a new column `CustomerNum` to the `Customers` table and assigns a default value of 88:

```
ALTER TABLE Customers
ADD CustomerNum INTEGER DEFAULT 88
```

- **Example 5** – moves **FP** indexes for `c2`, `c4`, and `c5`, from dbspace `Dsp3` to `Dsp6`. **FP** index for `c1` remains in `Dsp1`. **FP** index for `c3` remains in `Dsp2`. The primary key for `c5` remains in `Dsp4`. **DATE** index `c4_date` remains in `Dsp5`.

```
CREATE TABLE foo (
    c1 INT IN Dsp1,
    c2 VARCHAR(20),
    c3 CLOB IN Dsp2,
    c4 DATE,
    c5 BIGINT,
    PRIMARY KEY (c5) IN Dsp4) IN Dsp3);

CREATE DATE INDEX c4_date ON foo(c4) IN Dsp5;
ALTER TABLE foo
MOVE TO Dsp6;
```

- **Example 6** – moves only **FP** index `c1` from dbspace `Dsp1` to `Dsp7`:

```
ALTER TABLE foo ALTER c1 MOVE TO Dsp7
```

- **Example 7** – uses many **ALTER TABLE** clauses to move, split, rename, and merge partitions.

Create a partitioned table:

```
CREATE TABLE bar (
    c1 INT,
```

SQL Statements

```
        c2 DATE,  
        c3 VARCHAR(10))  
PARTITION BY RANGE(c2)  
  (p1 VALUES <= ('2005-12-31') IN dbbsp1,  
   p2 VALUES <= ('2006-12-31') IN dbbsp2,  
   P3 VALUES <= ('2007-12-31') IN dbbsp3,  
   P4 VALUES <= ('2008-12-31') IN dbbsp4);  
INSERT INTO bar VALUES(3, '2007-01-01', 'banana nut');  
INSERT INTO BAR VALUES(4, '2007-09-09', 'grape jam');  
INSERT INTO BAR VALUES(5, '2008-05-05', 'apple cake');
```

Move partition p2 to dbbsp5:

```
ALTER TABLE bar MOVE PARTITION p2 TO DBSP5;
```

Split partition p4 into 2 partitions:

```
ALTER TABLE bar SPLIT PARTITION p4 INTO  
  (P41 VALUES <= ('2008-06-30') IN dbbsp4,  
   P42 VALUES <= ('2008-12-31') IN dbbsp4);
```

This **SPLIT PARTITION** reports an error, as it requires data movement. Not all existing rows are in the same partition after split.

```
ALTER TABLE bar SPLIT PARTITION p3 INTO  
  (P31 VALUES <= ('2007-06-30') IN dbbsp3,  
   P32 VALUES <= ('2007-12-31') IN dbbsp3);
```

This error is reported:

```
No data move is allowed, cannot split partition p3.
```

This **SPLIT PARTITION** reports an error, because it changes the partition boundary value:

```
ALTER TABLE bar SPLIT PARTITION p2 INTO  
  (p21 VALUES <= ('2006-06-30') IN dbbsp2,  
   P22 VALUES <= ('2006-12-01') IN dbbsp2);
```

This error is reported:

```
Boundary value for the partition p2 cannot be changed.
```

Merge partition p3 into p2. An error is reported as a merge from a higher boundary value partition into a lower boundary value partition is not allowed.

```
ALTER TABLE bar MERGE PARTITION p3 into p2;
```

This error is reported:

```
Partition 'p2' is not adjacent to or before partition 'p3'.
```

Merge partition p2 into p3:

```
ALTER TABLE bar MERGE PARTITION p2 INTO P3;
```

Rename partition p1 to p1_new:

```
ALTER TABLE bar RENAME PARTITION p1 TO p1_new;
```

Unpartition table bar:

```
ALTER TABLE bar UNPARTITION;
```

Partition table bar. This command reports an error, because all rows must be in the first partition.

```
ALTER TABLE bar PARTITION BY RANGE (c2)
  (p1 VALUES <= ('2005-12-31') IN dbsp1,
   P2 VALUES <= ('2006-12-31') IN DBSP2,
   P3 VALUES <= ('2007-12-31') IN dbsp3,
   P4 VALUES <= ('2008-12-31') IN dbsp4);
```

This error is reported:

```
All rows must be in the first partition.
```

Partition table bar:

```
ALTER TABLE bar PARTITION BY RANGE (c2)
  (p1 VALUES <= ('2008-12-31') IN dbsp1,
   P2 VALUES <= ('2009-12-31') IN dbsp2,
   P3 VALUES <= ('2010-12-31') IN dbsp3,
   P4 VALUES <= ('2011-12-31') IN dbsp4);
```

- **Example 8** – changes a table `tab1` so that it is no longer registered for in-memory real-time updates in the RLV store.

```
ALTER TABLE tab1 DISABLE RLV STORE
```

Usage

The ALTER TABLE statement changes table attributes (column definitions and constraints) in a table that was previously created. The syntax allows a list of alter clauses; however, only one table constraint or column constraint can be added, modified, or deleted in each ALTER TABLE statement. ALTER TABLE is prevented whenever the statement affects a table that is currently being used by another connection. ALTER TABLE can be time consuming, and the server does not process requests referencing the same table while the statement is being processed.

Note: You cannot alter local temporary tables, but you can alter global temporary tables when they are in use by only one connection.

SAP Sybase IQ enforces REFERENCES and CHECK constraints. Table and/or column check constraints added in an ALTER TABLE statement are evaluated, only if they are defined on one of the new columns added, as part of that alter table operation. For details about CHECK constraints, see *CREATE TABLE Statement*.

If **SELECT *** is used in a view definition and you alter a table referenced by the **SELECT ***, then you must run **ALTER VIEW <viewname> RECOMPILE** to ensure that the view definition is correct and to prevent unexpected results when querying the view.

Side effects:

SQL Statements

- Automatic commit. The ALTER and DROP options close all cursors for the current connection. The Interactive SQL data window is also cleared.
- A checkpoint is carried out at the beginning of the ALTER TABLE operation.
- Once you alter a column or table, any stored procedures, views or other items that refer to the altered column no longer work.

Standards

- SQL – Vendor extension to ISO/ANSI SQL grammar.
- Sybase – Some clauses are supported by Adaptive Server Enterprise.

Permissions

Syntax 1

Requires one of:

- ALTER ANY TABLE system privilege
- ALTER ANY OBJECT system privilege
- ALTER privilege on the table
- You own the table

Syntax 2

The system privileges required for syntax 1 varies depending upon the clause used.

Clause	Privilege Required
Add	<p>Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER privilege on the underlying table • You own the underlying table <p>UNIQUE, PRIMARY KEY, FOREIGN KEY, or IQ UNIQUE column constraint – Requires above along with REFERENCE privilege on the underlying table.</p> <p>FOREIGN KEY table constraint requires above along with one of:</p> <ul style="list-style-type: none"> • CREATE ANY INDEX system privilege • CREATE ANY OBJECT system privilege • REFERENCE privilege on the base table <p>PARTITION BY RANGE requires above along with one of:</p> <ul style="list-style-type: none"> • CREATE ANY OBJECT system privilege • CREATE permission on the dbspaces where the partitions are being created
Alter	<p>Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table • You own the table. <p>To alter a primary key or unique constraint, also requires REFERENCE permission on the table.</p>

SQL Statements

Clause	Privilege Required
Drop	<p>Drop a column with no constraints – Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY OBJECT system privilege • ALTER ANY TABLE system privilege • ALTER permission on the underlying table • You own the underlying table <p>Drop a column or table with a constraint requires above along with REFERENCE permission if using ALTER permission.</p> <p>Drop a partition on table owned by self – None required.</p> <p>Drop a partition on table owned by other users – Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table
RENAME	<p>Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table • You own the table

Clause	Privilege Required
Move	<p>Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • MANAGE ANY DBSPACE system privilege • ALTER privilege on the underlying table • You own the underlying table <p>Also requires one of the following:</p> <ul style="list-style-type: none"> • CREATE ANY OBJECT system privilege • CREATE privilege on the dbspace to which the partition is being moved
Split Partition	<p>Partition on table owned by self – None required.</p> <p>Partition on table owned by other users – Requires one of:</p> <ul style="list-style-type: none"> • SELECT ANY TABLE system privilege • SELECT privilege on table <p>Also requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER privilege on the table
Merge Partition, Unpartition	<p>Table owned by self – None required.</p> <p>Table owned by other users – Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER privilege on the table

Clause	Privilege Required
Partition By	<p>Requires one of:</p> <ul style="list-style-type: none"> • CREATE ANY OBJECT system privilege • CREATE permission on the dbspaces where the partitions are being created <p>Also requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table • You own the table
Enable or disable RLV store	<p>Requires one of:</p> <ul style="list-style-type: none"> • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege

See also

- *CREATE TABLE Statement* on page 205
- *DROP Statement* on page 255
- *IDENTITY_INSERT Option* on page 551
- *FP_NBIT_AUTOSIZE_LIMIT Option* on page 536
- *FP_NBIT_ENFORCE_LIMITS Option* on page 538
- *FP_NBIT_LOOKUP_MB Option* on page 540
- *FP_NBIT_ROLLOVER_MAX_MB Option* on page 542

ALTER TEXT INDEX Statement

Renames, moves or alters the definition of a TEXT index.

Note: This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Syntax

```
ALTER TEXT INDEX [ owner. ] text-index-name
  ON [ owner. ] table-name
  alter-clause

alter-clause:
  rename-object | move-object
```

```

rename-object:
    RENAME { AS | TO } new-name

move-object:
    MOVE TO dbspace-name

```

Parameters

- **RENAME** – rename the TEXT index.
- **MOVE** – move the TEXT index to the specified dbspace.

Examples

- **Example** – creates a TEXT index, MyTextIndex, defining it as IMMEDIATE REFRESH, rename the TEXT index to Text_index_daily, and move the TEXT index to a dbspace named tispac:

```

CREATE TEXT INDEX MyTextIndex ON Customers ( CompanyName )
IMMEDIATE REFRESH;
ALTER TEXT INDEX MyTextIndex ON Customers RENAME AS
Text_index_daily;
ALTER TEXT INDEX Text_Index_Daily ON Customers MOVE TO tispac;

```

Usage

Side Effects:

- Automatic commit.

Permissions

move-object clause – Requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- REFERENCE permission on the underlying table.
- You own the underlying table

rename-object clause – Requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- MANAGE ANY DBSPACE
- One of the following:
 - You own the underlying table being indexed.
 - REFERENCE privilege on the table along with one of the following:
 - CREATE ANY OBJECT system privilege.

- CREATE privilege on the target dbspace.

ALTER TEXT CONFIGURATION Statement

Alters a text configuration object.

Note: This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Syntax

```
ALTER TEXT CONFIGURATION [ owner.]config-name
  STOPLIST stoplist
  | DROP STOPLIST
  | { MINIMUM | MAXIMUM } TERM LENGTH integer
  | TERM BREAKER
  | { GENERIC
    | [ EXTERNAL NAME library-and-entry-point-name-string ]
    | NGRAM }
  | PREFILTER EXTERNAL NAME library-and-entry-point-name-string
  | DROP PREFILTER
```

Parameters

- **stoplist** – a string-expression used to create or replace the list of terms to ignore when building a TEXT index. Terms specified in this list are also ignored in a query. Separate stoplist terms with spaces.

Stoplist terms cannot contain whitespace and should not contain non-alphanumeric characters. Non-alphanumeric characters are interpreted as spaces and break the term into multiple terms. For example, “and/or” is interpreted as the two terms “and” and “or”. The maximum number of stoplist terms is 7999.

- **DROP STOPLIST** – use to drop the stoplist for a text configuration object.
- **MINIMUM TERM LENGTH** – specifies the minimum length, in characters, of a term to include in the TEXT index. The value specified in the MINIMUM TERM LENGTH clause is ignored when using NGRAM TEXT indexes.

Terms that are shorter than this setting are ignored when building or refreshing the TEXT index. The value of this option must be greater than 0. If you set this option to be higher than MAXIMUM TERM LENGTH, the value of MAXIMUM TERM LENGTH is automatically adjusted to be the same as the new MINIMUM TERM LENGTH value.

- **MAXIMUM TERM LENGTH** – with GENERIC TEXT indexes, specifies the maximum length, in characters, of a term to include in the TEXT index. Terms that are longer than this setting are ignored when building or refreshing the TEXT index.

The value of MAXIMUM TERM LENGTH must be less than or equal to 60. If you set this option to be lower than MINIMUM TERM LENGTH, the value of MINIMUM TERM

LENGTH is automatically adjusted to be the same as the new MAXIMUM TERM LENGTH value.

- **TERM BREAKER** – specifies the name of the algorithm to use for separating column values into terms. The choices for IN SYSTEM tables are GENERIC (the default) or NGRAM. The GENERIC algorithm treats any string of one or more alphanumerics, separated by non-alphanumerics, as a term.

The NGRAM algorithm breaks strings into n-grams. An n-gram is an n-character substring of a larger string. The NGRAM term breaker is required for fuzzy (approximate) matching, or for documents that do not use whitespace or non-alphanumeric characters to separate terms. NGRAM is supported for IN SYSTEM tables.

NGRAM term breaker is built on TEXT indexes, so use text configuration object settings to define whether to use an NGRAM or GENERIC TEXT index.

TERM BREAKER can include the specification for the external term breaker library using EXTERNAL NAME and the library entry point.

- **library-and-entry-point-name-string** – *[operating-system:]function-name@library*
- **PREFILTER EXTERNAL NAME** – specifies the entry_point and the library name of the external pre-filter library provided by external vendors.
- **DROP PREFILTER** – drops the external prefilter and sets NULL to the prefilter columns in ISYSTEXTCONFIG table.

Examples

- **Example 1** – creates a text configuration object, maxTerm16, and then change the maximum term length to 16:

```
CREATE TEXT CONFIGURATION maxTerm16 FROM default_char;
ALTER TEXT CONFIGURATION maxTerm16 MAXIMUM TERM LENGTH 16;
```

- **Example 2** – adds stoplist terms to the maxTerm16 configuration object:

```
ALTER TEXT CONFIGURATION maxTerm16
STOPLIST 'because about therefore only';
```

- **Example 3** – updates the text configuration object, my_text_config, to use the entry point my_term_breaker in the external library mytermbreaker.dll for breaking the text:

```
CREATE TEXT CONFIGURATION my_text_config FROM default_char;
ALTER TEXT CONFIGURATION my_text_config
TERM BREAKER GENERIC EXTERNAL NAME
'platform:my_term_breaker@mytermbreaker';
```

- **Example 4** – updates the text configuration object, my_text_config, to use the entry point my_prefilter in the external library myprefilter.dll for prefiltering the documents:

SQL Statements

```
ALTER TEXT CONFIGURATION my_text_config  
PREFILTER EXTERNAL NAME 'platform:my_prefilter@myprefilter';
```

Usage

TEXT indexes are dependent on a text configuration object. SAP Sybase IQ TEXT indexes use immediate refresh, and cannot be truncated; you must drop the indexes before you can alter the text configuration object. To view the settings for text configuration objects, query the SYSTEXTCONFIG system view.

Side Effects:

- Automatic commit.

Permissions

TERM BREAKER or **PREFILTER EXTERNAL NAME** clause – Requires the CREATE ANY EXTERNAL REFERENCE system privilege, along with one of:

- ALTER ANY TEXT CONFIGURATION system privilege.
- ALTER ANY OBJECT system privilege.
- You own the text configuration object.

All other clauses – Requires the ALTER ANY TEXT CONFIGURATION system privilege, regardless of whether the user is the owner of the configuration object.

ALTER TRIGGER statement

Replaces a trigger definition with a modified version. You must include the entire new trigger definition in the ALTER TRIGGER statement.

Syntax 1 - Change the definition of a trigger

```
ALTER TRIGGER trigger-name trigger-definition
```

```
trigger-definition : CREATE TRIGGER syntax
```

Syntax 2 - Obfuscate a trigger definition

```
ALTER TRIGGER trigger-name ON [owner.] table-name SET HIDDEN
```

Remarks

- **Syntax 1** – The ALTER TRIGGER statement is identical in syntax to the CREATE TRIGGER statement except for the first word.

Either the Transact-SQL or Watcom SQL form of the CREATE TRIGGER syntax can be used.

- **Syntax 2** – You can use SET HIDDEN to obfuscate the definition of the associated trigger and cause it to become unreadable. The trigger can be unloaded and reloaded into other databases.

Note: The SET HIDDEN operation is irreversible.

Privileges

You must be the owner of the underlying table, or have the one of the following privileges:

ALTER privilege on the underlying table with the CREATE ANY OBJECT system privilege

ALTER ANY TRIGGER system privilege

ALTER ANY OBJECT system privilege

To alter a trigger on a view owned by someone else, you must have either the ALTER ANY TRIGGER and ALTER ANY VIEW system privileges, or you must have the ALTER ANY OBJECT system privilege.

Side effects

Automatic commit.

Standards and compatibility

- **SQL/2008** – Vendor extension.

ALTER USER Statement

Changes user settings.

Syntax

Syntax 1 – Change the definition of a database user

```
ALTER USER user-name
  | [ IDENTIFIED BY password ]
  | [ LOGIN POLICY policy-name ]
  | [ FORCE PASSWORD CHANGE { ON | OFF } ]
```

Syntax 2 – Refresh the Distinguished Name (DN) for an LDAP user

```
ALTER USER user-name
  REFRESH DN
```

Syntax 3 – Revert a user's login policy to the original values

```
ALTER USER user-name
  RESET LOGIN POLICY
```

Syntax 4 – Change a user's password when CHANGE_PASSWORD_DUAL_CONTROL is enabled in a user's login policy.

```
ALTER USER user-name  
IDENTIFIED [ FIRST | LAST ] BY password_part
```

Parameters

- **user-name** – name of the user.
- **IDENTIFIED BY** – the password for the user. Clause is not supported (ERROR) when CHANGE_PASSWORD_DUAL_CONTROL option is enabled in a user's login policy
- **IDENTIFIED[FIRST | LAST] BY** – clause mandatory when CHANGE_PASSWORD_DUAL_CONTROL option is enabled in a target user's login policy. FIRST | LAST keyword specifies the part of the dual password part being defined.
- **policy-name** – name of the login policy to assign the user. No change is made if you do not specify a login policy. No change is made if the LOGIN POLICY clause is not specified.
- **FORCE PASSWORD CHANGE** – controls whether the user must specify a new password upon logging in. This setting overrides the PASSWORD_EXPIRY_ON_NEXT_LOGIN option setting in the user's login policy.

Note: This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL).

- **RESET LOGIN POLICY** – reverts the settings of the user's login to the original values in the login policy. This usually clears all locks that are implicitly set due to the user exceeding the failed logins or exceeding the maximum number of days since the last login. When you reset a login policy, a user can access an account that has been locked for exceeding a login policy option limit such as MAX_FAILED_LOGIN_ATTEMPTS or MAX_DAYS_SINCE_LOGIN.
- **REFRESH DN** – clears the saved DN and timestamp for a user, which is used during LDAP authentication.

Examples

- **Example 1** – alters a user named SQLTester. The password is set to welcome. The SQLTester user is assigned to the Test1 login policy and the password does not expire on the next login:

```
ALTER USER SQLTester  
IDENTIFIED BY welcome  
LOGIN POLICY Test1  
FORCE PASSWORD CHANGE OFF
```

- **Example 2** – clears the distinguished name (DN) and timestamp for a user named Mary used for LDAP authentication:

```
ALTER USER Mary REFRESH DN
```

- **Example 3** – sets the password for user3 to PassPart1PassPart2. This assumes that user1 and user2 have the CHANGE PASSWORD system privilege and the change_password_dual_control option is enabled (ON) in the login policy for user3:

User1 enters:

```
ALTER USER user3 IDENTIFIED FIRST BY PassPart1
```

User2 enters:

```
ALTER USER user3 IDENTIFIED LAST BY PassPart2
```

Once set, user3 logs on by entering the password PassPart1PassPart2.

Usage

User IDs and passwords cannot:

- Begin with white space, single quotes, or double quotes
- End with white space
- Contain semicolons

Passwords cannot exceed 255 characters.

If you set the PASSWORD_EXPIRY_ON_NEXT_LOGIN value to ON, the passwords of all users assigned to this login policy expire immediately when he or she next logs in. You can use the **ALTER USER** and **LOGIN POLICY** clauses to force users to change their passwords at the next login.

If the CHANGE_PASSWORD_DUAL CONTROL login policy option is disable (OFF) during the dual password change process:

- the target user will be unable to log in with the single password part already defined. The **ALTER USER** command must be reissued using single password control syntax.
- If the option is disabled after the dual password change process is complete, but before the target user logs in, there is no impact on the target user. The target user must log in using both password parts.

If the target user is already logged in when the dual password change process occurs, the user cannot change their password in the current session until both parts of the new password are set. Once the dual password change process is complete, the target user can use **GRANT CONNECT**, **ALTER USER**, **sp_password**, or **sp_iqpassword** to the password without first logging out. The prompt to enter the current password, use the new dual control password, not the password originally entered for the current session.

The **GRANT CONNECT** statement is not supported during for the dual password change process to set either password part. However, once the dual password change process is complete, the target user can use the **GRANT CONNECT** statement, **ALTER USER**, **sp_password**, or **sp_iqpassword** to change their password without first logging out.

SQL Statements

As soon as both parts of the password are successfully specified by users with the CHANGE PASSWORD system privilege, the password for the target user is automatically expired. This forces the target user to change the password the next time he or she logs in.

The encryption algorithm used for hashing the user passwords is FIPS-certified encryption support:

- The DLL is called **dbfips10.dll**
- The HASH function accepts the algorithms: **SHA1_FIPS** **SHA256_FIPS**
- If the **-fips** server option is specified and an algorithm that is not FIPS-certified is given to the HASH function, the database server uses **SHA1_FIPS** instead of **SHA1**, **SHA256_FIPS** instead of **SHA256**, and returns an error if **MD5** is used (**MD5** is not a FIPS-certified algorithm).
- If the **-fips** option is specified, the database server uses **SHA256_FIPS** for password hashing.

Standards

- SQL – Vendor extension to ISO/ANSI SQL grammar.
- Sybase – Not supported by Adaptive Server Enterprise.

Permissions

- To change own password – None required.
- To change the password of any user – Requires the CHANGE PASSWORD system privilege.
- To use the **LOGIN POLICY**, **FORCE PASSWORD CHANGE**, **RESET LOGIN POLICY**, or **REFRESH DN** clauses requires the **MANAGE ANY USER** system privilege.

See also

- *COMMENT Statement* on page 96
- *CREATE LOGIN POLICY Statement* on page 155
- *CREATE USER Statement* on page 230
- *DROP LOGIN POLICY Statement* on page 263
- *DROP USER Statement* on page 275
- *ALTER LOGIN POLICY Statement* on page 24
- *GRANT ROLE Statement* on page 312
- *GRANT System Privilege Statement* on page 319
- *REVOKE System Privilege Statement* on page 408
- *REVOKE ROLE Statement* on page 404

ALTER VIEW Statement

Replaces a view definition with a modified version.

Syntax

Syntax 1 – Alter the structure of the view

ALTER VIEW

```
... [ owner.]view-name [ ( column-name [ , ... ] ) ]
... AS select-statement
... [ WITH CHECK OPTION ]
```

Syntax 2 – Change attributes for the view

ALTER VIEW

```
... [ owner.]view-name
... { SET HIDDEN | RECOMPILE | DISABLE | ENABLE }
```

Parameters

- **AS** – the **SELECT** statement on which the view is based must not contain an **ORDER BY** clause, a subquery in the **SELECT** list, or a **TOP** or **FIRST** qualification. It may have a **GROUP BY** clause and may be a **UNION**.
- **WITH CHECK OPTION** – rejects any updates and inserts to the view that do not meet the criteria of the views as defined by its **SELECT** statement. However, SAP Sybase IQ currently ignores this option (it supports the syntax for compatibility reasons).
- **SET HIDDEN** – obfuscate the definition of the view and cause the view to become hidden from view, for example in Sybase Control Center. Explicit references to the view still work.

Warning! The **SET HIDDEN** operation is irreversible.

When you use **SET HIDDEN**, you can unload and reload the view into other databases. Debugging using the debugger does not show the view definition, nor is it available through procedure profiling. If you need to change the definition of a hidden view, you must drop the view and create it again using the **CREATE VIEW** statement.

- **RECOMPILE** – recreate the column definitions for the view. Identical in functionality to the **ENABLE** clause, except you can use it on a view that is not disabled.
- **DISABLE** – disable the view from use by the database server.

When you use the **DISABLE** clause, the view is no longer available for use by the database server to answer queries. Disabling a view is similar to dropping one, except that the view definition remains in the database. Disabling a view also disables any dependent views.

SQL Statements

Therefore, the **DISABLE** clause requires exclusive access, not only to the view being disabled, but to any dependent views, which are also disabled.

- **ENABLE** – enable a disabled view, which causes the database server to recreate the column definitions for the view. Before you enable a view, you must enable any views on which it depends.

Usage

When you alter a view, existing permissions on the view are maintained and do not require reassignment. Instead of using the **ALTER VIEW** statement, you could also drop the view and recreate it using **DROP VIEW** and **CREATE VIEW**, respectively. If you do this, view permissions must be reassigned.

After completing the view alteration using Syntax 1, the database server recompiles the view. Depending on the type of change you made, if there are dependent views, the database server attempts to recompile them. If you made changes that impact a dependent view, that view may become invalid, requiring you to alter the definition for the dependent view.

Warning! If the **SELECT** statement defining the view contains an asterisk (*), the number of the columns in the view could change if columns were added or deleted from the underlying tables. The names and data types of the view columns could also change.

Altering the structure of a view requires that you replace the entire view definition with a new definition, much as you would when creating the view using the **CREATE VIEW** statement.

Side effects:

- Automatic commit
- All procedures and triggers are unloaded from memory, so that any procedure or trigger that references the view reflects the new view definition. The unloading and loading of procedures and triggers can have a performance impact if you regularly alter views.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

RECOMPILE or **ENABLE** clause – For view requires one of:

- **ALTER ANY VIEW** system privilege.
- **ALTER ANY OBJECT** system privilege.
- You own the view.
- Also require one of:
 - **SELECT ANY TABLE** system privilege.

- `SELECT` privilege on the underlying tables of the view.

For materialized view requires one of:

- `ALTER ANY MATERIALIZED VIEW` system privilege.
- `ALTER ANY OBJECT` system privilege.
- You own the materialized view.
- Also require one of:
 - `SELECT ANY TABLE` system privilege.
 - `SELECT` privilege on the underlying tables of the materialized view.

DISABLE clause – For view requires one of:

- `ALTER ANY VIEW` system privilege.
- `ALTER ANY OBJECT` system privilege.
- You own the view.

For materialized view requires one of:

- `ALTER ANY MATERIALIZED VIEW` system privilege.
- `ALTER ANY OBJECT` system privilege.
- You own the materialized view.

All other clauses require one of:

- `ALTER ANY OBJECT` system privilege.
- You own the view.

See also

- *CREATE VIEW Statement* on page 234
- *DROP Statement* on page 255
- *Identifying and Fixing Invalid Dependent Views* on page 77

Identifying and Fixing Invalid Dependent Views

Check for, and correct, any dependent views that become invalid due to changes to their underlying tables.

Under most circumstances the database server automatically recompiles views to keep them valid if the underlying tables change. However, if your table alteration removes or materially changes something referenced by the view definition, then the dependent view becomes invalid. For example, if you remove a column referenced in the view definition, then the dependent view is no longer valid. Correct the view definition and manually recompile the view.

1. Run `sa_dependent_views` to get the list of dependent views.

SQL Statements

2. Perform the DDL operation that alters the table. The server automatically disables dependent views, and attempts to recompile them once the DDL is complete.
3. Check that all the views listed by **sa_dependent_views** are valid. For example, perform a simple test such as **SELECT * FROM myview**.
4. If a view is invalid, it is likely you will need to alter the view definition to resolve the issue. Examine the view definition against the DDL change that you made and make the necessary changes. Run **ALTER VIEW RECOMPILE** to correct the view definition.
5. Test the corrected view to make sure it works. For example, perform a simple test such as **SELECT * FROM myview**.

The **sa_dependent_views** system procedure returns the list of all dependent views for a given table or view.

See also

- *ALTER VIEW Statement* on page 75

BACKUP Statement

Backs up an SAP Sybase IQ database on one or more archive devices.

Syntax

BACKUP DATABASE

```
[ backup-option... ]  
TO archive_device [ archive-option... ]  
... [ WITH COMMENT string ]
```

backup-option:

```
{ READWRITE FILES ONLY |  
  READONLY dbspace-or-file [, ... ] }  
CRC { ON | OFF }  
ATTENDED { ON | OFF }  
BLOCK FACTOR integer  
{ FULL | INCREMENTAL | INCREMENTAL SINCE FULL }  
VIRTUAL { DECOUPLED |  
  ENCAPSULATED 'shell_command' }  
WITH COMMENT comment
```

dbspace-or-file:

```
{ DBSPACES identifier-list | FILES identifier-list }
```

identifier-list:

```
identifier [, ... ]
```

archive-option:

```
SIZE integer STACKER integer
```


Parameters

- **TO** – specify the name of the archive_device to be used for backup, delimited with single quotation marks. The archive_device is a file name or tape drive device name for the archive file. If you use multiple archive devices, specify them using separate TO clauses. (A comma-separated list is not allowed.) Archive devices must be distinct. The number of TO clauses determines the amount of parallelism SAP Sybase IQ attempts with regard to output devices.
- **WITH COMMENT** – specify an optional comment recorded in the archive file and in the backup history file. Maximum length is 32KB. If you do not specify a value, a NULL string is stored.
- **READWRITE FILES ONLY** – restricts FULL, INCREMENTAL, and INCREMENTAL SINCE FULL backups to only the set of read-write files in the database. The read-write dbspaces/files must be SAP Sybase IQ dbspaces.

If READWRITE FILES ONLY clause is used with an INCREMENTAL or INCREMENTAL SINCE FULL backup, the backup will not back up data on read-only dbspaces or dbfiles that has changed since the depends-on backup. If READWRITE FILES ONLY is not specified for an INCREMENTAL or INCREMENTAL SINCE FULL backup, the backup backs up all database pages that have changed since the depends-on backup, both on read-write and read-only dbspaces.

- **CRC** – activates 32-bit cyclical redundancy checking on a per block basis (in addition to whatever error detection is available in the hardware). When you specify this clause, the numbers computed on backup are verified during any subsequent RESTORE operation, affecting performance of both commands. The default is ON.
- **ATTENDED** – applies only when backing up to a tape device. If ATTENDED ON clause (the default) is used, a message is sent to the application that issued the **BACKUP** statement if the tape drive requires intervention. This might happen, for example, when a new tape is required. If you specify OFF, **BACKUP** does not prompt for new tapes. If additional tapes are needed and OFF has been specified, SAP Sybase IQ gives an error and aborts the **BACKUP** command. However, a short delay is included to account for the time an automatic stacker drive requires to switch tapes.
- **BLOCK FACTOR *integer*** – specify the number of blocks to write at one time. The value must be greater than 0, or SAP Sybase IQ generates an error message. Its default is 25 for UNIX systems and 15 for Windows systems (to accommodate the smaller fixed tape block sizes). This clause effectively controls the amount of memory used for buffers. The actual amount of memory is this value times the block size times the number of threads used to extract data from the database. Set BLOCK FACTOR to at least 25.
- **FULL | INCREMENTAL | INCREMENTAL SINCE FULL** –
 - **FULL** – specify a full backup; all blocks in use in the database are saved to the archive devices. This is the default action.

- **INCREMENTAL** – specify an incremental backup; all blocks changed since the last backup of any kind are saved to the archive devices. The keyword **INCREMENTAL** is not allowed with **READONLY FILES**.
- **INCREMENTAL SINCE FULL** – specify an incremental backup; all blocks changed since the last full backup are saved to the archive devices.
- **VIRTUAL DECOUPLED** – specify a decoupled virtual backup. For the backup to be complete, you must copy the SAP Sybase IQ dbspaces after the decoupled virtual backup finishes, and then perform a nonvirtual incremental backup.
- **VIRTUAL ENCAPSULATED** – specify an encapsulated virtual backup. The 'shell-command' argument can be a string or variable containing a string that is executed as part of the encapsulated virtual backup. The shell commands execute a system-level backup of the IQ store as part of the backup operation. For security reasons, it is recommended that an absolute path be specified in the 'shell-command,' and file protections on that directory be in place to prevent execution of an unintended program.
- **SIZE clause** – Specify maximum tape or file capacity per output device (some platforms do not reliably detect end-of-tape markers). No volume used on the corresponding device should be shorter than this value. This value applies to both tape and disk files but not third-party devices. Units are kilobytes (KB), although in general, less than 1GB is inappropriate. For example, for a 3.5GB tape, specify 3500000. Defaults are by platform and medium. The final size of the backup file will not be exact, because backup writes in units of large blocks of data.

Table 1. BACKUP default sizes

Platform	Default SIZE for Tape	Default SIZE for Disk
UNIX	none	2GB
Windows	1.5GB SIZE must be a multiple of 64. Other values are rounded down to a multiple of 64.	1.5GB

The **SIZE** parameter is per output device. **SIZE** does not limit the number of bytes per device; **SIZE** limits the file size. Each output device can have a different **SIZE** parameter. During backup, when the amount of information written to a given device reaches the value specified by the **SIZE** parameter, **BACKUP** does one of the following:

- If the device is a file system device, **BACKUP** closes the current file and creates another file of the same name, with the next ascending number appended to the file name, for example, bkup1.dat1.1, bkup1.dat1.2, bkup1.dat1.3.
- If the device is a tape unit, **BACKUP** closes the current tape and you need to mount another tape.
- **STACKER** – specify that the device is automatically loaded, and specifies the number of tapes with which it is loaded. This value is not the tape position in the stacker, which could be zero. When **ATTENDED** is **OFF** and **STACKER** is **ON**, SAP Sybase IQ waits for a predetermined amount of time to allow the next tape to be autoloaded. The number of tapes

supplied along with the **SIZE** clause are used to determine whether there is enough space to store the backed-up data. Do not use this clause with third-party media management devices.

Examples

- **Example 1** – this UNIX example backs up the `iqdemo` database onto tape devices `/dev/rmt/0` and `/dev/rmt/2` on a Sun Solaris platform. On Solaris, the letter `n` after the device name specifies the “no rewind on close” feature. Always specify this feature with **BACKUP**, using the naming convention appropriate for your UNIX platform (Windows does not support this feature). This example backs up all changes to the database since the last full backup:

```
BACKUP DATABASE
INCREMENTAL SINCE FULL
TO '/dev/rmt/0n' SIZE 10000000
TO '/dev/rmt/2n' SIZE 15000000
```

Note: Size units are kilobytes (KB), although in most cases, size of less than 1GB are inappropriate. In this example, the specified sizes are 10GB and 15GB.

- **Example 2** – these **BACKUP** commands specify read-only files and dbspaces:

```
BACKUP DATABASE READONLY DBSPACES dsp1
TO '/dev/rmt/0'

BACKUP DATABASE READONLY FILES dsp1_f1, dsp1_f2
TO 'bkp.f1f2'

BACKUP DATABASE READONLY DBSPACES dsp2, dsp3
READONLY FILES dsp4_f1, dsp5_f2
TO 'bkp.RO'
```

Usage

The SAP Sybase IQ database might be open for use by many readers and writers when you execute a **BACKUP** command. It acts as a read-only user and relies on the Table Level Versioning feature of SAP Sybase IQ to achieve a consistent set of data.

BACKUP implicitly issues a **CHECKPOINT** prior to commencing, and then it backs up the catalog tables that describe the database (and any other tables you have added to the catalog store). During this first phase, SAP Sybase IQ does not allow any metadata changes to the database (such as adding or dropping columns and tables). Correspondingly, a later **RESTORE** of the backup restores only up to that initial **CHECKPOINT**.

The **BACKUP** command lets you specify full or incremental backups. You can choose two kinds of incremental backups. **INCREMENTAL** backup only those blocks that have changed and committed since the last **BACKUP** of any type (incremental or full). **INCREMENTAL SINCE FULL** backs up all of the blocks that have changed since the last full backup. The first type of incremental backup can be smaller and faster to do for **BACKUP** commands, but slower and more complicated for **RESTORE** commands. The opposite is true for the other type of

incremental backup. The reason is that the first type generally results in N sets of incremental backup archives for each full backup archive. If a restore is required, a user with the **SERVER OPERATOR** system privilege must **RESTORE** the full backup archive first, and then each incremental archive in the proper order. (SAP Sybase IQ keeps track of which ones are needed.) The second type requires the user with the **SERVER OPERATOR** system privilege to restore only the full backup archive and the last incremental archive.

Incremental virtual backup is supported using the **VIRTUAL DECOUPLED** and **VIRTUAL ENCAPSULATED** parameters of the **BACKUP** statement.

Although you can perform an OS-level copy of tablespaces to make a virtual backup of one or more read-only dbspaces, use the virtual backup statement, because it records the backup in the SAP Sybase IQ system tables.

BACKUP and **RESTORE** write your SAP Sybase IQ data in parallel to or from all of the archive devices you specify. The catalog store is written serially to the first device. Faster backups and restores result from greater parallelism.

SAP Sybase IQ supports a maximum of 36 hardware devices for backup. For faster backups, specifying one or two devices per core will help to avoid hardware and IO contention. Set the **SIZE** parameter on the **BACKUP** command to avoid creating multiple files per backup device and consider the value used in the **BLOCK FACTOR** clause on the **BACKUP** command.

BACKUP overwrites existing archive files unless you move the old files or use a different *archive_device* name or path.

The backup API DLL implementation lets you specify arguments to pass to the DLL when opening an archive device. For third-party implementations, the *archive_device* string has this format:

```
'DLLidentifier::vendor_specific_information'
```

A specific example:

```
'spsc::workorder=12;volname=ASD002'
```

The *archive_device* string length can be up to 1023 bytes. The *DLLidentifier* portion must be 1 to 30 bytes in length and can contain only alphanumeric and underscore characters. The *vendor_specific_information* portion of the string is passed to the third-party implementation without checking its contents. Do not specify the **SIZE** or **STACKER** clauses of the **BACKUP** command when using third-party implementations, as that information should be encoded in the *vendor_specific_information* portion of the string.

Note: Only certain third-party products are certified with SAP Sybase IQ using this syntax. See the *Release Bulletin* for additional usage instructions or restrictions. Before using any third-party product to back up your SAP Sybase IQ database in this way, make sure it is certified. See the *Release Bulletin*, or see the SAP Sybase Certification Reports for the SAP Sybase IQ product in *Technical Documents* at <http://www.sybase.com/support/techdocs/>.

For the Sybase implementation of the backup API, you need to specify only the tape device name or file name. For disk devices, you should also specify the `SIZE` value, or SAP Sybase IQ assumes that each created disk file is no larger than 2GB on UNIX, or 1.5GB on Windows.

An example of an archive device for the SAP Sybase API DLL that specifies a tape device for certain UNIX systems is:

```
' /dev/rmt/0 '
```

It is your responsibility to mount additional tapes if needed, or to ensure that the disk has enough space to accommodate the backup.

When multiple devices are specified, **BACKUP** distributes the information across all devices. Other issues for **BACKUP** include:

- **BACKUP** does not support raw devices as archival devices.
- Windows systems support only fixed-length I/O operations to tape devices (for more information about this limitation, see your *Installation and Configuration Guide*). Although Windows supports tape partitioning, SAP Sybase IQ does not use it, so do not use another application to format tapes for **BACKUP**. Windows has a simpler naming strategy for its tape devices, where the first tape device is `\\.|tape0`, the second is `\\.|tape1`, and so on.

Warning! For backup (and for most other situations) SAP Sybase IQ treats the leading backslash in a string as an escape character, when the backslash precedes an `n`, an `x`, or another backslash. For this reason, when you specify backup tape devices, you must double each backslash required by the Windows naming convention. For example, indicate the first Windows tape device you are backing up to as `'\\\\. \\tape0'`, the second as `'\\\\. \\tape1'`, and so on. If you omit the extra backslashes, or otherwise misspell a tape device name, and write a name that is not a valid tape device on your system, SAP Sybase IQ interprets this name as a disk file name.

- SAP Sybase IQ does not rewind tapes before using them. You must ensure the tapes used for **BACKUP** or **RESTORE** are at the correct starting point before putting them in the tape device. SAP Sybase IQ does rewind tapes after using them on rewinding devices.
- During **BACKUP** and **RESTORE** operations, if SAP Sybase IQ cannot open the archive device (for example, when it needs the media loaded) and the `ATTENDED` clause is `ON`, it waits for ten seconds and tries again. It continues these attempts indefinitely until either it is successful or the operation is terminated with a `Ctrl+C`.
- If you enter `Ctrl+C`, **BACKUP** fails and returns the database to the state it was in before the backup started.
- If disk striping is used, such as on a RAID device, the striped disks are treated as a single device.

Side effects:

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires one of:

- BACK UP DATABASE system privilege.
- You own the database.

See also

- *RESTORE DATABASE Statement* on page 389

BEGIN ... END Statement

Groups SQL statements together.

Syntax

```
[ statement-label : ]
... BEGIN [ [ NOT ] ATOMIC ]
... [ local-declaration ; ... ]
... statement-list
... [ EXCEPTION [ exception-case ... ] ]
... END [ statement-label ]

local-declaration:
{ variable-declaration
| cursor-declaration
| exception-declaration
| temporary-table-declaration }

variable-declaration:
DECLARE variable-name [ , ... ] data-type
[ { = | DEFAULT } initial-value ]

initial-value:
special-value
| string
| [ - ] number
| ( constant-expression )
| built-in-function ( constant-expression )
| NULL

special-value:
CURRENT {
    DATABASE
    | DATE
    | PUBLISHER
```

```

| TIME
| TIMESTAMP
| USER
| UTC TIMESTAMP }
| USER

```

Parameters

- **statement-label** – if specified, it must match the beginning *statement-label*. You can use the **LEAVE** statement to resume execution at the first statement after the compound statement. The compound statement that is the body of a procedure has an implicit label that is the same as the name of the procedure.
- **initial-value** – if specified, the variable is set to that value and the data type must match the type defined by *data-type*. If you do not specify an initial-value, the variable contains the NULL value until a **SET** statement assigns a different value.

Examples

- **Example 1** – the body of a procedure is a compound statement:

```

CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION FOR
        SQLSTATE '02000' ;
    DECLARE curThisCust CURSOR FOR
        SELECT CompanyName, CAST(
            sum(SalesOrderItems.Quantity *
                Products.UnitPrice) AS INTEGER) VALUE
        FROM Customers
            LEFT OUTER JOIN Salesorders
            LEFT OUTER JOIN SalesOrderItems
            LEFT OUTER JOIN Products
        GROUP BY CompanyName ;
    DECLARE ThisValue INT ;
    DECLARE ThisCompany CHAR(35) ;
    SET TopValue = 0 ;
    OPEN curThisCust ;

    CustomerLoop:
    LOOP
        FETCH NEXT curThisCust
            INTO ThisCompany, ThisValue ;
        IF SQLSTATE = err_notfound THEN
            LEAVE CustomerLoop ;
        END IF ;
        IF ThisValue > TopValue THEN
            SET TopValue = ThisValue ;
            SET TopCompany = ThisCompany ;
        END IF ;
    END LOOP CustomerLoop ;

```

```
CLOSE curThisCust ;  
END
```

Usage

The body of a procedure is a compound statement. Compound statements can also be used in control statements within a procedure.

A compound statement allows one or more **SQL** statements to be grouped together and treated as a unit. A compound statement starts with **BEGIN** and ends with **END**. Immediately after **BEGIN**, a compound statement can have local declarations that exist only within the compound statement. A compound statement can have a local declaration for a variable, a cursor, a temporary table, or an exception. Local declarations can be referenced by any statement in that compound statement, or in any compound statement nested within it. Local declarations are invisible to other procedures that are called from within a compound statement.

An atomic statement is a statement executed completely or not at all. For example, an **UPDATE** statement that updates thousands of rows might encounter an error after updating many rows. If the statement does not complete, all changes revert back to their original state. Similarly, if you specify that the **BEGIN** statement is atomic, the statement is executed either in its entirety or not at all.

Standards

- **SQL**—Vendor extension to ISO/ANSI SQL grammar.
- **Sybase**—Supported by Adaptive Server Enterprise. This does not mean that all statements inside a compound statement are supported.

BEGIN and **END** keywords are not required in Transact-SQL.

BEGIN and **END** are used in Transact-SQL to group a set of statements into a single compound statement, so that control statements such as **IF ... ELSE**, which affect the performance of only a single **SQL** statement, can affect the performance of the whole group. The **ATOMIC** keyword is not supported by Adaptive Server Enterprise.

In Transact-SQL, **DECLARE** statements need not immediately follow **BEGIN**, and the cursor or variable that is declared exists for the duration of the compound statement. You should declare variables at the beginning of the compound statement for compatibility.

Permissions

None

See also

- *DECLARE LOCAL TEMPORARY TABLE Statement* on page 246
- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *LEAVE Statement* on page 342

- *RESIGNAL Statement* on page 388
- *SIGNAL Statement* on page 437

BEGIN PARALLEL IQ ... END PARALLEL IQ Statement

Groups **CREATE INDEX** statements together for execution at the same time.

Syntax

```
... BEGIN PARALLEL IQ
   statement-list
... END PARALLEL IQ
```

Parameters

- **statement-list** – a list of **CREATE INDEX** statements

Examples

- **Example 1** – this statement executes atomically. If one command fails, the entire statement rolls back:

```
BEGIN PARALLEL IQ
  CREATE HG INDEX c1_HG on table1 (col1);
  CREATE HNG INDEX c12_HNG on table1 (col12);
  CREATE LF INDEX c1_LF on table1 (col1);
  CREATE HNG INDEX c2_HNG on table1 (col2);
END PARALLEL IQ
```

Usage

The **BEGIN PARALLEL IQ ... END PARALLEL IQ** statement lets you execute a group of **CREATE INDEX** statements as though they are a single DDL statement, creating indexes on multiple IQ tables at the same time. While this statement is executing, you and other users cannot issue other DDL statements.

Note:

- This statement does not support RLV-enabled tables.
 - This statement does not support TEXT indexes.
-

Side effects:

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise. For support of statements inside the statement, see *CREATE INDEX Statement*.

Permissions

None

See also

- *CREATE INDEX Statement* on page 142

BEGIN TRANSACTION Statement [T-SQL]

Use this statement to begin a user-defined transaction.

Note: **BEGIN TRANSACTION** is a T-SQL construct and must contain only valid T-SQL commands. You cannot mix T-SQL and non-T-SQL commands.

Syntax

```
BEGIN TRAN [SACTION] [ transaction-name ]
```

Examples

- **Example 1** – reports successive values of @@trancount as 0, 1, 2, 1, 0 and prints the values on the server window:

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
```

Do not rely on the value of @@trancount for more than keeping track of the number of explicit **BEGIN TRANSACTION** statements that have been issued.

When Adaptive Server Enterprise starts a transaction implicitly, the @@trancount variable is set to 1. SAP Sybase IQ does not set the @@trancount value to 1 when a transaction is started implicitly. So, the SAP Sybase IQ @@trancount variable has a value of zero before any **BEGIN TRANSACTION** statement (even though there is a current transaction), while in Adaptive Server Enterprise (in chained mode) it has a value of 1.

For transactions starting with a **BEGIN TRANSACTION** statement, @@trancount has a value of 1 in both SAP Sybase IQ and Adaptive Server Enterprise after the first **BEGIN**

TRANSACTION statement. If a transaction is implicitly started with a different statement, and a **BEGIN TRANSACTION** statement is then executed, @@trancount has a value of 2 in both SAP Sybase IQ, and Adaptive Server Enterprise after the **BEGIN TRANSACTION** statement.

Usage

The optional parameter *transaction-name* is the name assigned to this transaction. It must be a valid identifier. Use transaction names only on the outermost pair of nested **BEGIN/COMMIT** or **BEGIN/ROLLBACK** statements.

When executed inside a transaction, the **BEGIN TRANSACTION** statement increases the nesting level of transactions by one. The nesting level is decreased by a **COMMIT** statement. When transactions are nested, only the outermost **COMMIT** makes the changes to the database permanent.

Both Adaptive Server Enterprise and SAP Sybase IQ have two transaction modes.

The default Adaptive Server Enterprise transaction mode, called unchained mode, commits each statement individually, unless an explicit **BEGIN TRANSACTION** statement is executed to start a transaction. In contrast, the ISO SQL/2003 compatible chained mode only commits a transaction when an explicit **COMMIT** is executed or when a statement that carries out an autocommit (such as data definition statements) is executed.

You can control the mode by setting the chained database option. The default setting for ODBC and embedded SQL connections in SAP Sybase IQ is On, in which case SAP Sybase IQ runs in chained mode. (ODBC users should also check the AutoCommit ODBC setting). The default for TDS connections is Off.

In unchained mode, a transaction is implicitly started before any data retrieval or modification statement. These statements include: **DELETE**, **INSERT**, **OPEN**, **FETCH**, **SELECT**, and **UPDATE**. You must still explicitly end the transaction with a **COMMIT** or **ROLLBACK** statement.

You cannot alter the chained option within a transaction.

Note: When calling a stored procedure, you should ensure that it operates correctly under the required transaction mode.

The current nesting level is held in the global variable @@trancount. The @@trancount variable has a value of zero before the first **BEGIN TRANSACTION** statement is executed, and only a **COMMIT** executed when @@trancount is equal to one makes changes to the database permanent.

A **ROLLBACK** statement without a transaction or savepoint name always rolls back statements to the outermost **BEGIN TRANSACTION** (explicit or implicit) statement, and cancels the entire transaction.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

None

See also

- *COMMIT Statement* on page 102
- *ROLLBACK TRANSACTION Statement [T-SQL]* on page 414
- *SAVE TRANSACTION Statement [T-SQL]* on page 416
- *ISOLATION_LEVEL Option* on page 561

CALL Statement

Invokes a procedure.

Syntax

Syntax 1

```
[ variable = ] CALL procedure-name ( [ expression ] [ , ... ] )
```

Syntax 2

```
[ variable = ] CALL procedure-name ( [ parameter-name = expression ]  
[ , ... ] )
```

Examples

- **Example 1** – calls the `sp_customer_list` procedure. This procedure has no parameters, and returns a result set:

```
CALL sp_customer_list()
```

- **Example 2** – creates a procedure to return the number of orders placed by the customer whose ID is supplied, creates a variable to hold the result, calls the procedure, and displays the result:

```
CREATE PROCEDURE OrderCount (IN CustomerID INT, OUT Orders INT)  
BEGIN  
SELECT COUNT("DBA".SalesOrders.ID)  
INTO Orders  
FROM "DBA".Customers  
KEY LEFT OUTER JOIN "DBA".SalesOrders  
WHERE "DBA".Customers.ID = CustomerID ;  
END  
go
```

```
-- Create a variable to hold the result
CREATE VARIABLE Orders INT
go

-- Call the procedure, FOR customer 101
-----
CALL OrderCount ( 101, Orders)
go
-----
-- Display the result
SELECT Orders FROM DUMMY
go
```

Usage

CALL invokes a procedure that has been previously created with a **CREATE PROCEDURE** statement. When the procedure completes, any INOUT or OUT parameter values are copied back.

You can specify the argument list by position or by using keyword format. By position, arguments match up with the corresponding parameter in the parameter list for the procedure. By keyword, arguments match the named parameters.

Procedure arguments can be assigned default values in the **CREATE PROCEDURE** statement, and missing parameters are assigned the default value, or, if no default is set, NULL.

Inside a procedure, **CALL** can be used in a **DECLARE** statement when the procedure returns result sets.

Note: You cannot reference a Table UDF in a **CALL SQL** statement.

Procedures can return an integer value (as a status indicator, say) using the **RETURN** statement. You can save this return value in a variable using the equality sign as an assignment operator:

```
CREATE VARIABLE returnval INT ;
returnval = CALL proc_integer ( arg1 = val1, ... )
```

Note: Use of this statement to invoke a function is deprecated. To call functions, use an assignment statement to invoke the function and assign its result to a variable. For example:

```
DECLARE varname INT;
SET varname=test( );
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise. For an alternative that is supported, see *EXECUTE Statement [ESQL]*.

Permissions

Requires one of:

- EXECUTE ANY PROCEDURE system privilege.
- EXECUTE permission for the procedure
- You own the procedure

See also

- *CREATE PROCEDURE Statement* on page 165
- *EXECUTE Statement [ESQL]* on page 276
- *GRANT EXECUTE Statement* on page 310

CASE Statement

The **CASE** statement is a control statement that lets you choose a list of SQL statements to execute based on the value of an expression.

Syntax

```
CASE value-expression
...WHEN [ constant | NULL ] THEN statement-list ...
... [ WHEN [ constant | NULL ] THEN statement-list ] ...
...ELSE statement-list
... END
```

Examples

- **Example 1** – classifies the products listed in the `Products` table of the demo database into one of shirt, hat, shorts, or unknown:

```
CREATE PROCEDURE ProductType (IN product_id INT, OUT type
CHAR(10))
BEGIN
  DECLARE prod_name CHAR(20) ;
  SELECT name INTO prod_name FROM "GROUPO"."Products"
  WHERE ID = product_id;
  CASE prod_name
  WHEN 'Tee Shirt' THEN
    SET type = 'Shirt'
  WHEN 'Sweatshirt' THEN
    SET type = 'Shirt'
  WHEN 'Baseball Cap' THEN
    SET type = 'Hat'
  WHEN 'Visor' THEN
    SET type = 'Hat'
  WHEN 'Shorts' THEN
    SET type = 'Shorts'
  ELSE
    SET type = 'UNKNOWN'
```

```
END CASE ;
END
```

Usage

If a WHEN clause exists for the value of *value-expression*, the *statement-list* in the WHEN clause is executed. If no appropriate WHEN clause exists, and an ELSE clause exists, the *statement-list* in the ELSE clause is executed. Execution resumes at the first statement after the END.

Note: The ANSI standard allows two forms of **CASE** statements. Although SAP Sybase IQ allows both forms, when **CASE** is in the predicate, for best performance you must use the form shown here.

If you require the other form (also called ANSI syntax) for compatibility with SQL Anywhere, use this syntax:

```
CASE
  WHEN [ search-condition | NULL] THEN statement-list ...
  [ WHEN [ search-condition | NULL] THEN statement-list ] ...
  [ ELSE statement-list ]
END [ CASE ]
```

With this ANSI syntax form, the statements are executed for the first satisfied search-condition in the **CASE** statement. The ELSE clause is executed if none of the *search-conditions* are met. If the expression can be NULL, use the following syntax for the first *search-condition*:

```
WHEN search-condition IS NULL THEN statement-list
```

Attention: Do not confuse the syntax of the **CASE** statement with that of the CASE expression.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84

CHECKPOINT Statement

Checkpoints the database.

Syntax

CHECKPOINT

Usage

CHECKPOINT forces the database server to execute a checkpoint. Checkpoints are also performed automatically by the database server according to an internal algorithm. Applications do not normally need to issue **CHECKPOINT**.

SAP Sybase IQ uses checkpoints differently than OLTP databases such as SQL Anywhere. OLTP databases tend to have short transactions that affect only a small number of rows. Writing entire pages to disk would be very expensive for them. Instead, OLTP databases generally write to disk at checkpoints, and write only the changed data rows. SAP Sybase IQ is an OLAP database. A single OLAP transaction can change thousands or millions of rows of data. For this reason, the database server does not wait for a checkpoint to occur to perform physical writes. It writes updated data pages to disk after each transaction commits. For an OLAP database, writing full pages of data to disk is much more effective than writing small amounts of data at arbitrary checkpoints.

Adjusting the checkpoint time or issuing explicit checkpoints may be unnecessary. Controlling checkpoints is less important in SAP Sybase IQ than in OLTP database products, because SAP Sybase IQ writes the actual data pages after each transaction commits.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

Requires the CHECKPOINT system privilege.

CLEAR Statement [Interactive SQL]

Closes any open result sets in Interactive SQL (**dbisql**).

Syntax

CLEAR

Usage

Closes any open result sets and leaves the contents of the SQL Statements pane unchanged

Side effects:

The **CLEAR** statement closes the cursor associated with the data being cleared.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *EXIT Statement [Interactive SQL]* on page 282

CLOSE Statement [ESQL] [SP]

Closes a named cursor.

Syntax

```
CLOSE cursor-name
```

```
cursor-name:
  { identifier | host-variable }
```

Examples

- **Example 1** – closes cursors in Embedded SQL:

```
EXEC SQL CLOSE employee_cursor;
EXEC SQL CLOSE :cursor_var;
```

- **Example 2** – uses a cursor:

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
  DECLARE err_notfound EXCEPTION
  FOR SQLSTATE '02000' ;
  DECLARE curThisCust CURSOR FOR
  SELECT CompanyName,
  CAST(      sum(SalesOrderItems.Quantity *
  Products.UnitPrice) AS INTEGER) VALUE
  FROM Customers
  LEFT OUTER JOIN SalesOrders
```

SQL Statements

```
    LEFT OUTER JOIN SalesOrderItems
    LEFT OUTER JOIN Products
GROUP BY CompanyName ;
DECLARE ThisValue INT ;
DECLARE ThisCompany CHAR(35) ;
SET TopValue = 0 ;
OPEN curThisCust ;
CustomerLoop:
LOOP
    FETCH NEXT curThisCust
    INTO ThisCompany, ThisValue ;
    IF SQLSTATE = err_notfound THEN
        LEAVE CustomerLoop ;
    END IF ;
    IF ThisValue > TopValue THEN
        SET TopValue = ThisValue ;
        SET TopCompany = ThisCompany ;
    END IF ;
END LOOP CustomerLoop ;
CLOSE curThisCust ;
END
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

The cursor must have been previously opened.

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *OPEN Statement [ESQL] [SP]* on page 369
- *PREPARE Statement [ESQL]* on page 375

COMMENT Statement

Stores a comment, in the system tables, about a database object.

Syntax

```
COMMENT ON
{ COLUMN [ owner.]table-name.column-name
| DBSPACE dbspace-name
| EVENT event-name
| EXTERNAL [ENVIRONMENT] OBJECT object-name
| EXTERNAL ENVIRONMENT environment-name
| EXTERNAL OBJECT object-name
| FOREIGN KEY [owner.]table-name.role-name
```

```

| INDEX [ [owner.]table.]index-name
| INTEGRATED LOGIN integrated-login-id
| JAVA CLASS java-class-name
| JAVA JAR java-jar-name
| KERBEROS LOGIN "client-Kerberos-principal"
| LDAP SERVER ldap-server-name
| LOGICAL SERVER logical-server-name
| LOGIN POLICY policy-name
| LS POLICY ls-policy-name
| MATERIALIZED VIEW [owner.]materialized-view-name
| PRIMARY KEY ON [owner.]table-name
| PROCEDURE [owner.]table-name
| ROLE role-name
| SERVICE web-service-name
| SEQUENCE [owner.]sequence-name
| SPATIAL REFERENCE SYSTEM srs-name
| SPATIAL UNIT OF MEASURE uom-identifier
| TABLE [ owner.]table-name
| TEXT CONFIGURATION [ owner.]text-config-name
| TEXT INDEX text-index-name
| TRIGGER [[owner.]table-name.]trigger-name
| USER userid
| VIEW [ owner.]view-name
}
IS comment

environment-name:
    JAVA | PERL | PHP | C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64

comment:
    { string | NULL }

```

Examples

- **Example 1** – adds a comment to the Employees table:

```

COMMENT
ON TABLE Employees
IS "Employee information"

```

- **Example 2** – removes the comment from the Employees table:

```

COMMENT
ON TABLE Employees
IS NULL

```

Usage

The **COMMENT** statement updates remarks in the ISYSREMARK system table. You can remove a comment by setting it to NULL. The owner of a comment on an index or trigger is the owner of the table on which the index or trigger is defined.

The **COMMENT ON DBSPACE**, **COMMENT ON JAVA JAR**, and **COMMENT ON JAVA CLASS** statements allow you to set the Remarks column in the SYS.ISYSREMARK system table. Remove a comment by setting it to NULL.

SQL Statements

You cannot add comments for local temporary tables.

Note: Materialized views are supported only for SQL Anywhere tables in the IQ catalog store.

Standards

- SQL – Vendor extension to ISO/ANSI SQL grammar.
- Sybase – Not supported by Adaptive Server Enterprise.

Permissions

Clause	Privilege Required
COLUMN	Any one of: <ul style="list-style-type: none">• You own the table• CREATE ANY TABLE system privilege• ALTER ANY TABLE system privilege• CREATE ANY OBJECT system privilege• ALTER ANY OBJECT system privilege• COMMENT ANY OBJECT system privilege
DBSPACE	MANAGE ANY DBSPACE system privilege
EVENT	Any one of: <ul style="list-style-type: none">• MANAGE ANY EVENT• CREATE ANY OBJECT• ALTER ANY OBJECT• COMMENT ANY OBJECT
EXTERNAL [ENVIRONMENT] OBJECT	MANAGE ANY EXTERNAL OBJECT system privilege
EXTERNAL ENVIRONMENT	MANAGE ANY EXTERNAL ENVIRONMENT system privilege
FOREIGN KEY	Any one of: <ul style="list-style-type: none">• You own the table• CREATE ANY TABLE system privilege• ALTER ANY TABLE system privilege• CREATE ANY OBJECT system privilege• ALTER ANY OBJECT system privilege• COMMENT ANY OBJECT system privilege

Clause	Privilege Required
INDEX	Any one of: <ul style="list-style-type: none"> • You own the index • CREATE ANY INDEX system privilege • ALTER ANY INDEX system privilege • COMMENT ANY OBJECT system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege
INTEGRATED LOGIN	MANAGE ANY USER system privilege
JAVA CLASS or JAVA JAR	MANAGE ANY EXTERNAL OBJECT system privilege
KERBEROS LOGIN	MANAGE ANY USER system privilege
LDAP SERVER	MANAGE ANY LDAP SERVER system privilege
LOGICAL SERVER	MANAGE MULTIPLEX system privilege
LOGIN POLICY	MANAGE ANY LOGIN POLICY system privilege
LS POLICY	MANAGE MULTIPLEX system privilege
MATERIALIZED VIEW	Any one of: <ul style="list-style-type: none"> • You own the view • CREATE ANY MATERIALIZED VIEW system privilege • ALTER ANY MATERIALIZED VIEW system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
PRIMARY KEY ON	Any one of: <ul style="list-style-type: none"> • You own the table • CREATE ANY TABLE system privilege • ALTER ANY TABLE system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege

Clause	Privilege Required
PROCEDURE	Any one of: <ul style="list-style-type: none"> • You own the procedure • CREATE ANY PROCEDURE system privilege • ALTER ANY PROCEDURE system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
SEQUENCE	Any one of: <ul style="list-style-type: none"> • You own the sequence • CREATE ANY SEQUENCE system privilege • ALTER ANY SEQUENCE system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
SERVICE	MANAGE ANY WEB SERVICE system privilege
SPATIAL REFERENCE SYSTEM	Any one of: <ul style="list-style-type: none"> • COMMENT ANY OBJECT • CREATE ANY OBJECT • ALTER ANY OBJECT • MANAGE ANY SPATIAL OBJECT
SPATIAL UNIT OF MEASURE	Any one of: <ul style="list-style-type: none"> • COMMENT ANY OBJECT • CREATE ANY OBJECT • ALTER ANY OBJECT • MANAGE ANY SPATIAL OBJECT
ROLE	System role – administrative privilege over the role being commented on. User-defined role – MANAGE ROLES system privilege or administrative privilege over the role being commented on.

Clause	Privilege Required
TABLE	Any one of: <ul style="list-style-type: none"> • You own the table • CREATE ANY TABLE system privilege • ALTER ANY TABLE system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
TEXT CONFIGURATION	Any one of: <ul style="list-style-type: none"> • You created the text configuration • CREATE ANY TEXT CONFIGURATION system privilege • ALTER ANY TEXT CONFIGURATION system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
TEXT INDEX	Any one of: <ul style="list-style-type: none"> • You created the text index • CREATE ANY INDEX system privilege • ALTER ANY INDEX system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
TRIGGER	Any one of: <ul style="list-style-type: none"> • You created the trigger • CREATE ANY TRIGGER system privilege • ALTER ANY TRIGGER system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege
USER	MANAGE ANY USER system privilege

Clause	Privilege Required
VIEW	Any one of: <ul style="list-style-type: none"> • You own the view • CREATE ANY VIEW system privilege • ALTER ANY VIEW system privilege • CREATE ANY OBJECT system privilege • ALTER ANY OBJECT system privilege • COMMENT ANY OBJECT system privilege

COMMIT Statement

Makes changes to the database permanent, or terminates a user-defined transaction.

Syntax

Syntax 1 – To end a transaction and makes all changes permanent

```
COMMIT [ WORK ]
```

Syntax 2 – To construct nested transactions

```
COMMIT TRAN[SACTION ] [ transaction-name ]
```

Examples

- **Example 1** – commits the current transaction:

```
COMMIT
```

- **Example 2** – this Transact-SQL batch reports successive values of @@trancount as 0, 1, 2, 1, 0:

```
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
BEGIN TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
COMMIT TRANSACTION
PRINT @@trancount
go
```

Usage

- **Syntax 1** – Data definition statements carry out commits automatically. For information, see the *Side effects* listing for each SQL statement.

COMMIT fails if the database server detects any invalid foreign keys. This makes it impossible to end a transaction with any invalid foreign keys. Usually, foreign key

integrity is checked on each data manipulation operation. However, if the database option `WAIT_FOR_COMMIT` is set `ON` or a particular foreign key was defined with a **CHECK ON COMMIT** clause, the database server delays integrity checking until the **COMMIT** statement is executed.

- **Syntax 2** – Nested transactions are similar to savepoints. When executed as the outermost of a set of nested transactions, the statement makes changes to the database permanent. When executed inside a transaction, **COMMIT TRANSACTION** decreases the nesting level of transactions by one. When transactions are nested, only the outermost **COMMIT** makes the changes to the database permanent.

The optional parameter *transaction-name* is the name assigned to this transaction. It must be a valid identifier. Use transaction names only on the outermost pair of nested **BEGIN/COMMIT** or **BEGIN/ROLLBACK** statements.

You can use a set of options to control the detailed behavior of the **COMMIT** statement. See *COOPERATIVE_COMMIT_TIMEOUT Option*, *COOPERATIVE_COMMITS Option*, *DELAYED_COMMITS Option*, and *DELAYED_COMMIT_TIMEOUT Option*. You can use the **Commit** connection property to return the number of commits on the current connection.

Side effects:

- Closes all cursors except those opened `WITH HOLD`.
- Deletes all rows of declared temporary tables on this connection, unless they were declared using `ON COMMIT PRESERVE ROWS`.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Adaptive Server Enterprise. Syntax 2 is a Transact-SQL extension to ISO/ANSI SQL grammar.

Permissions

Must be connected to the database.

See also

- *BEGIN TRANSACTION Statement [T-SQL]* on page 88
- *CONNECT Statement [ESQL] [Interactive SQL]* on page 104
- *DISCONNECT Statement [Interactive SQL]* on page 254
- *ROLLBACK Statement* on page 413
- *SAVEPOINT Statement* on page 415
- *SET CONNECTION Statement [ESQL] [Interactive SQL]* on page 429
- *COOPERATIVE_COMMIT_TIMEOUT Option* on page 506
- *COOPERATIVE_COMMITS Option* on page 507

- *DELAYED_COMMITS* Option on page 525
- *DELAYED_COMMIT_TIMEOUT* Option on page 525

CONFIGURE Statement [Interactive SQL]

Activates the Interactive SQL (**dbisql**) configuration window.

Syntax

```
CONFIGURE
```

Usage

The **dbisql** configuration window displays the current settings of all **dbisql** options. It does not display or let you modify database options.

If you select Permanent, the options are written to the `SYSOPTION` table in the database and the database server performs an automatic **COMMIT**. If you do not choose Permanent, and instead click OK, options are set temporarily and remain in effect for the current database connection only.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

None

See also

- *SET OPTION* Statement on page 431

CONNECT Statement [ESQL] [Interactive SQL]

Establishes a connection to the database identified by *database-name* running on the server identified by *engine-name*.

Syntax

Syntax 1

```
CONNECT
```

```
... [ TO engine-name ]  
...[ DATABASE database-name ]
```

```
...[ AS connection-name ]
...[ USER ] userid [ IDENTIFIED BY ]
```

Syntax 2

```
CONNECT USING connect-string
```

Parameters

- **engine-name** – identifier, string, or host-variable
- **database-name** – identifier, string, or host-variable
- **connection-name** – identifier, string, or host-variable
- **userid** – identifier, string, or host-variable
- **password** – identifier, string, or host-variable
- **connect-string** – a list of parameter settings of the form `keyword=value`, and must be enclosed in single quotes.
- **AS** – connection can optionally be named by specifying the clause. This allows multiple connections to the same database, or multiple connections to the same or different database servers, all simultaneously. Each connection has its own associated transaction. You might even get locking conflicts between your transactions if, for example, you try to modify the same record in the same database from two different connections.

Examples

- **Example 1 – CONNECT usage within Embedded SQL:**

```
EXEC SQL CONNECT AS :conn_name
USER :userid IDENTIFIED BY :password;
EXEC SQL ] CONNECT USER "dba" IDENTIFIED BY "sql";
```

- **Example 2 – CONNECT usage from dbisql:**

Connect to a database from **dbisql**. Prompts display for user ID and password:

```
CONNECT
```

Connect to the default database as DBA, from **dbisql**. A password prompt displays:

```
CONNECT USER "DBA"
```

Connect to the demo database as the DBA, from **dbisql**, where `<machine>_iqdemo` is the engine name:

```
CONNECT
TO <machine>_iqdemo
USER "DBA"
IDENTIFIED BY sql
```

Connect to the demo database using a connect string, from **dbisql**:

```
CONNECT  
USING 'UID=DBA;PWD=sql;DBN=iqdemo'
```

Usage

- **Embedded SQL behavior** – in Embedded SQL, if no *engine-name* is specified, the default local database server is assumed (the first database server started). If no *database-name* is specified, the first database on the given server is assumed.

The **WHENEVER** statement, **SET SQLCA**, and some **DECLARE** statements do not generate code and thus might appear before the **CONNECT** statement in the source file. Otherwise, no statements are allowed until a successful **CONNECT** statement has been executed.

The user ID and password are used for permission checks on all dynamic SQL statements. By default, the password is case-sensitive; the user ID is not. You can connect without a password by using a host variable for the password and setting the value of the host variable to be the null pointer.

- **dbisql behavior** – if no database or server is specified in the **CONNECT** statement, **dbisql** remains connected to the current database, rather than to the default server and database. If a database name is specified without a server name, **dbisql** attempts to connect to the specified database on the current server. You must specify the database name defined in the **-n** database switch, not the database file name. If a server name is specified without a database name, **dbisql** connects to the default database on the specified server. For example, if this batch is executed while connected to a database, the two tables are created in the same database.

```
CREATE TABLE t1 ( c1 int );  
CONNECT DBA IDENTIFIED BY sql;  
CREATE TABLE t2 (c1 int );
```

No other database statements are allowed until a successful **CONNECT** statement has been executed.

The user ID and password are used for checking the permissions on SQL statements. If the password or the user ID and password are not specified, the user is prompted to type the missing information. By default, the password is case-sensitive; the user ID is not.

Multiple connections are managed through the concept of a current connection. After a successful connect statement, the new connection becomes the current one. To switch to a different connection, use **SET CONNECTION**. Executing a **CONNECT** statement does not close the existing connection (if any). Use **DISCONNECT** to drop connections.

Static SQL statements use the user ID and password specified with the **-l** option on the **SQLPP** statement line. If no **-l** option is given, then the user ID and password of the **CONNECT** statement are used for static SQL statements also.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.

- Sybase—Open Client Embedded SQL supports a different syntax for the **CONNECT** statement.

Permissions

None

See also

- *DISCONNECT Statement [Interactive SQL]* on page 254
- *GRANT CONNECT Statement* on page 306
- *SET CONNECTION Statement [ESQL] [Interactive SQL]* on page 429

CREATE DATABASE Statement

Creates a database consisting of several operating system files.

Syntax

```
CREATE DATABASE db-name
... [ [ TRANSACTION ] { LOG ON [ log-file-name ]
      [ MIRROR mirror-file-name ] } ]
... [ CASE { RESPECT | IGNORE } ]
... [ PAGE SIZE catalog-page-size ]
... [ COLLATION collation-label[( collation-tailoring-string ) ] ]
... [ ENCRYPTED {algorithm-key-spec | OFF } ]
... [ BLANK PADDING ON ]
... [ JCONNECT { ON | OFF } ]
... [ IQ PATH iq-file-name ]
... [ IQ SIZE iq-file-size ]
... [ IQ PAGE SIZE iq-page-size ]
... [ BLOCK SIZE block-size ]
... [ IQ RESERVE sizeMB ]
... [ TEMPORARY RESERVE sizeMB ]
... [ MESSAGE PATH message-file-name ]
... [ TEMPORARY PATH temp-file-name ]
... [ TEMPORARY SIZE temp-db-size ]
... [ DBA USER userid ]
... [ DBA PASSWORD password ]
... [ SYSTEM PROCEDURE AS DEFINER {ON | OFF} ]
```

file-name:

```
db-name
| log-file-name
| mirror-file-name
| iq-file-name
| message-file-name
| temp-file-name
```

catalog-page-size (bytes):

```
{ 4096 | 8192 | 16384 | 32768 }
```

SQL Statements

```
iq-page-size (bytes):
  { 65536 | 131072 | 262144 | 524288 }

block-size (bytes):
  { 4096 | 8192 | 16384 | 32768 }

collation-label:
  string

collation-tailoring-string:
  keyword=value

collation-tailoring-string:
  keyword=value

algorithm-key-spec:
  ON
  | [ ON ] KEY key [ ALGORITHM AES-algorithm ]
  | [ ON ] ALGORITHM AES-algorithm KEY key
  | [ ON ] ALGORITHM 'SIMPLE'

AES-algorithm:
  'AES' | 'AES256' | 'AES_FIPS' | 'AES256_FIPS'

key:
  quoted string
```

Parameters

- **TRANSACTION LOG** – a file where the database server logs all changes made to the database. The transaction log plays a key role in system recovery. If you do not specify any TRANSACTION LOG clause, or if you omit a path for the file name, it is placed in the same directory as the .db file. However, you should place it on a different physical device from the .db and .iq. It cannot be created on a raw partition
- **MIRROR** – an identical copy of a transaction log, usually maintained on a separate device, for greater protection of your data. By default, SAP Sybase IQ does not use a mirrored transaction log. If you do want to use a transaction log mirror, you must provide a file name. If you use a relative path, the transaction log mirror is created relative to the directory of the catalog store (db-name.db). Tip: Always create a mirror copy of the transaction log.
- **CASE** – for databases created with CASE RESPECT, all affected values are case-sensitive in comparisons and string operations. Database object names such as columns, procedures, or user IDs, are unaffected. Dbspace names are always case-insensitive, regardless of the CASE specification. The default (RESPECT) is that all comparisons are case-sensitive. CASE RESPECT provides better performance than CASE IGNORE.
- **PAGE SIZE** – page size for the SQL Anywhere segment of the database (containing the catalog tables) can be 4096, 8192, 16384, or 32768 bytes. Normally, use the default, 4096 (4KB). Large databases might need a larger page size than the default and may see performance benefits as a result. The smaller values might limit the number of columns

your database can support. If you specify a page size smaller than 4096, SAP Sybase IQ uses a page size of 4096.

- **COLLATION** – the collation sequence used for sorting and comparison of character data types in the database. The collation provides character comparison and ordering information for the encoding (character set) being used. If the **COLLATION** clause is not specified, SAP Sybase IQ chooses a collation based on the operating system language and encoding. For most operating systems, the default collation sequence is **ISO_BINENG**, which provides the best performance. In **ISO_BINENG**, the collation order is the same as the order of characters in the ASCII character set. All uppercase letters precede all lowercase letters (for example, both ‘A’ and ‘B’ precede ‘a’).

You can choose the collation from a list of supported collations. For SQL Anywhere databases created on an SAP Sybase IQ server, the collation can also be the Unicode Collation Algorithm (UCA). If UCA is specified, also specify the **ENCODING** clause. SAP Sybase IQ does not support any of the UCA-based collations for IQ databases. If a UCA-based collation is specified in the **CREATE DATABASE** statement for a database, the server returns the error `UCA collation is not supported` and database creation fails. A collation sequence cannot be changed after the database is created.

Optionally, you can specify collation tailoring options (*collation-tailoring-string*) for additional control over the sorting and comparing of characters. These options take the form of keyword=value pairs, assembled in parentheses, following the collation name.

Collation tailoring options for SAP Sybase IQ contains the supported keyword, allowed alternate forms, and allowed values for the collation tailoring option (*collation-tailoring-string*) for an SAP Sybase IQ database.

Table 2. Collation Tailoring Option for SAP Sybase IQ

Keyword	Colla-tion	Alternate Forms	Allowed Values
CaseSensi-tivity	All suppor-ted colla-tions	CaseSensi-tive, Case	<ul style="list-style-type: none"> • respect – respect case differences between letters. For the UCA collation, this is equivalent to UpperFirst. For other collations, the value of respect depends on the collation itself. • ignore – ignore case differences between letters. • UpperFirst – always sort upper case first (Aa). • LowerFirst – always sort lowercase first (aA).

Note: Several collation tailoring options are supported when you specify the UCA collation for a SQL Anywhere database created on an SAP Sybase IQ server. For all other collations and for SAP Sybase IQ, only case sensitivity tailoring is supported. Also,

databases created with collation tailoring options cannot be started using a pre-15.0 database server.

- **ENCRYPTED** – makes the data stored in your physical database file unreadable. Use the **CREATE DATABASE ENCRYPTED** keyword without the **TABLE** keyword to encrypt the entire database. Use the **ENCRYPTED TABLE** clause to enable only table encryption for SQL Anywhere tables. Table-level encryption is not supported for SAP Sybase IQ tables. Enabling table encryption means that the tables that are subsequently created or altered using the **ENCRYPTED** clause are encrypted using the settings you specified at database creation.

There are two levels of database encryption: simple and strong.

- Simple encryption is equivalent to obfuscation. The data is unreadable, but someone with cryptographic expertise could decipher the data. For simple encryption, specify the **CREATE DATABASE** clause **ENCRYPTED ON ALGORITHM 'SIMPLE'**, **ENCRYPTED ALGORITHM 'SIMPLE'**, or specify the **ENCRYPTED ON** clause without specifying an algorithm or key.
- Strong encryption is achieved through the use of a 128-bit algorithm and a security key. The data is unreadable and virtually undecipherable without the key. For strong encryption, specify the **CREATE DATABASE** clause **ENCRYPTED ON ALGORITHM** with a 128-bit or 256-bit AES algorithm and use the **KEY** clause to specify an encryption key. You should choose a value for your key that is at least 16 characters long, contains a mix of uppercase and lowercase, and includes numbers, letters, and special characters.

This encryption key is required each time you start the database.

You can specify encryption only during database creation. To introduce encryption to an existing database requires a complete unload, database re-creation, and reload of all data. If the **ENCRYPTED** clause is used but no algorithm is specified, the default is AES. By default, encryption is OFF.

Warning! Protect your encryption key! Store a copy of your key in a safe location. A lost key results in a completely inaccessible database from which there is no recovery.

- **BLANK PADDING** – trailing blanks are ignored for comparison purposes (**BLANK PADDING ON**), and Embedded SQL programs pad strings that are fetched into character arrays. This option is provided for compatibility with the ISO/ANSI SQL standard. **CREATE DATABASE** no longer supports **BLANK PADDING OFF**.
- **JCONNECT** – to use the SAP Sybase jConnect for JDBC driver to access system catalog information, install jConnect support. Set **JCONNECT** to **OFF** to exclude the jConnect system objects (the default is **ON**). You can still use JDBC, as long as you do not access system information.
- **IQ PATH** – the path name of the main segment file containing the SAP Sybase IQ data. You can specify an operating system file or a raw partition of an I/O device. (The Installation and Configuration Guide guide for your platform describes the format for

specifying a raw partition.) SAP Sybase IQ automatically detects which type based on the path name you specify. If you use a relative path, the file is created relative to the directory of the catalog store (the .db file).

If you omit the IQ PATH clause, specifying any of these options generates an error: IQ SIZE, IQ PAGE SIZE, BLOCK SIZE, MESSAGE PATH, TEMPORARY PATH, and TEMPORARY SIZE.

- **IQ SIZE** – the size in MB of either the raw partition or the operating system file you specify with the IQ PATH clause. For raw partitions, you should always take the default by not specifying IQ SIZE, which allows SAP Sybase IQ to use the entire raw partition; if you specify a value for IQ SIZE, the value must match the size of the I/O device or SAP Sybase IQ returns an error. For operating system files, you can specify a value from the minimum in the following table up to a maximum of 4TB.

The default size for an operating system file depends on IQ PAGE SIZE:

Table 3. Default and Minimum Sizes of IQ and Temporary Store Files

IQ PAGE SIZE	IQ SIZE Default	TEMPORARY SIZE Default	Minimum Explicit IQ SIZE	Minimum Explicit TEMPORARY SIZE
65536	4096000	2048000	4MB	2MB
131072	8192000	4096000	8MB	4MB
262144	16384000	8192000	16MB	8MB
524288	32768000	16384000	32MB	16MB

- **IQ PAGE SIZE** – the page size, in bytes, for the SAP Sybase IQ segment of the database (containing the IQ tables and indexes). The value must be a power of 2, from 65536 to 524288 bytes. The default is 131072 (128KB). Other values for the size are changed to the next larger size. The IQ page size determines the default I/O transfer block size and maximum data compression for your database.

For best performance, use these minimum page sizes:

- 64KB (IQ PAGE SIZE 65536) for databases whose largest table contains up to 1 billion rows, or a total size less than 8TB. This is the absolute minimum for a new database. On 32-bit platforms, a 64KB IQ page size gives the best performance.
- 128KB (IQ PAGE SIZE 131072) for databases on a 64-bit platform whose largest table contains more than 1 billion rows and fewer than 4 billion rows, or might grow to a total size of 8TB or greater. 128KB is the default IQ page size.
- 256KB (IQ PAGE SIZE 262144) for databases on a 64-bit platform whose largest table contains more than 4 billion rows, or might grow to a total size of 8TB or greater.
- **BLOCK SIZE** – the I/O transfer block size, in bytes, for the SAP Sybase IQ segment of the database. The value must be less than IQ PAGE SIZE, and must be a power of two

between 4096 and 32768. Other values for the size are changed to the next larger size. The default value depends on the value of the IQ PAGE SIZE clause. For most applications, the default value is optimum.

- **IQ RESERVE** – size, in megabytes, of space to reserve for the main IQ store (IQ_SYSTEM_MAIN dbspace), so that the dbfile can be increased in size in the future. The sizeMB parameter can be any number greater than 0. You cannot change the reserve after the dbspace is created. When IQ RESERVE is specified, the database uses more space for internal (free list) structures. If reserve size is too large, the space needed for the internal structures can be larger than the specified size, which results in an error.
- **TEMPORARY RESERVE** – size, in megabytes, of space to reserve for the temporary IQ store (IQ_SYSTEM_TEMP dbspace), so that the dbfile can be increased in size in the future. The sizeMB parameter can be any number greater than 0. You cannot change the reserve after the dbspace is created. When TEMPORARY RESERVE is specified, the database uses more space for internal (free list) structures. If reserve size is too large, the space needed for the internal structures can be larger than the specified size, which results in an error.

Note: Reserve and mode for temporary dbspaces are lost if the database is restored from a backup.

- **MESSAGE PATH** – path name of the segment containing the SAP Sybase IQ messages trace file. You must specify an operating system file; the message file cannot be on a raw partition. If you use a relative path or omit the path, the message file is created relative to the directory of the .db file.
- **TEMPORARY SIZE** – size, in megabytes, of either the raw partition or the operating system file you specify with the TEMPORARY PATH clause. For raw partitions, always use the default by not specifying TEMPORARY SIZE, which allows SAP Sybase IQ to use the entire raw partition. The default for operating system files is always one-half the value of IQ SIZE. If the IQ store is on a raw partition and the temporary store is an operating system file, the default TEMPORARY SIZE is half the size of the IQ store raw partition.
- **DBA USER** – user name for the default user account granted the SYS_AUTH_DBA_ROLE system role. If you do not specify this clause, SAP Sybase IQ creates a default DBA user ID.
- **DBA PASSWORD** – password for the default user account granted the SYS_AUTH_DBA_ROLE system role.
- **SYSTEM PROCEDURE AS DEFINER** – defines whether a privileged system procedure runs with the privileges of the invoker (the person executing the procedure) or the definer (the owner of the procedure). OFF (default), or not specified, means all privileged system procedures execute with the privileges of the invoker. Use sp_proc_priv() to identify the system privileges required to run a system procedure.

ON means that pre-16.0 privileged system procedures execute with the privileges of the definer. 16.0 or later privileged system procedures execute with the privileges of the invoker.

Examples

- **Example 1** – this Windows example creates an SAP Sybase IQ database named `mydb` with its corresponding `mydb.db`, `mydb.iq`, `mydb.iqtmp`, and `mydb.iqmsg` files in the `C:\s1\data` directory:

```
CREATE DATABASE 'C:\\s1\\data\\mydb'
BLANK PADDING ON
IQ PATH 'C:\\s1\\data'
IQ SIZE 2000
IQ PAGE SIZE 131072
```

- **Example 2** – this UNIX command creates an SAP Sybase IQ database with raw devices for `IQ PATH` and `TEMPORARY PATH`. The default IQ page size of 128KB applies.

```
CREATE DATABASE '/s1/data/bigdb'
IQ PATH '/dev/md/rdisk/bigdb'
MESSAGE PATH '/s1/data/bigdb.iqmsg'
TEMPORARY PATH '/dev/md/rdisk/bigtmp'
```

- **Example 3** – this Windows command creates an SAP Sybase IQ database with a raw device for `IQ PATH`. Note the doubled backslashes in the raw device name (a Windows requirement):

```
CREATE DATABASE 'company'
IQ PATH '\\\\.\\E:'
JCONNECT OFF
IQ SIZE 40
```

- **Example 4** – this UNIX example creates a strongly encrypted SAP Sybase IQ database using the AES encryption algorithm with the key “is!seCret.”

```
CREATE DATABASE 'marvin.db'
BLANK PADDING ON
CASE RESPECT
COLLATION 'ISO_BINENG'
IQ PATH '/filesystem/marvin.main1'
IQ SIZE 6400
IQ PAGE SIZE 262144
TEMPORARY PATH '/filesystem/marvin.temp1'
TEMPORARY SIZE 3200
ENCRYPTED ON KEY 'is!seCret' ALGORITHM 'AES'
```

Usage

Creates a database with the supplied name and attributes. The `IQ PATH` clause is required for creating the SAP Sybase IQ database; otherwise, you create a standard SQL Anywhere database.

When SAP Sybase IQ creates a database, it automatically generates four database files to store different types of data that constitute a database. Each file corresponds to a dbspace, the logical name by which SAP Sybase IQ identifies database files:

- *db-name.db* is the file that holds the catalog dbspace, SYSTEM. It contains the system tables and stored procedures describing the database and any standard SQL Anywhere database objects you add. If you do not include the .db extension, SAP Sybase IQ adds it. This initial dbspace contains the catalog store, and you can later add dbspaces to increase its size. It cannot be created on a raw partition.
- *db-name.iq* is the default name of the file that holds the main data dbspace, IQ_SYSTEM_MAIN, which contains the IQ tables and indexes. You can specify a different file name with the IQ PATH clause. This initial dbspace contains the IQ store.

Warning! IQ_SYSTEM_MAIN is a special dbspace that contains all structures necessary for the database to open: the IQ db_identity blocks, the IQ checkpoint log, the IQ rollforward/rollback bitmaps of each committed transaction and each active checkpointed transaction, the incremental backup bitmaps, and the freelist root pages.

IQ_SYSTEM_MAIN is always online when the database is open.

The administrator can allow user tables to be created in IQ_SYSTEM_MAIN, especially if these tables are small, important tables. However, it is more common that immediately after creating the database, the administrator creates a second main dbspace, revokes create privilege in dbspace IQ_SYSTEM_MAIN from all users, grants create privilege on the new main dbspace to selected users, and sets PUBLIC.default_dbspace to the new main dbspace.

-
- *db-name.iqtmp* is the default name of the file that holds the initial temporary dbspace, IQ_SYSTEM_TEMP. It contains the temporary tables generated by certain queries. The required size of this file can vary depending on the type of query and amount of data. You can specify a different name using the TEMPORARY PATH clause. This initial dbspace contains the temporary store.
 - *db-name.iqmsg* is the default name of the file that contains the messages trace dbspace, IQ_SYSTEM_MSG. You can specify a different file name using the MESSAGE PATH clause.

In addition to these files, a database has a transaction log file (*db-name.log*), and might have a transaction log mirror file.

File names and the **CREATE DATABASE** statement:

The file names (*db-name*, *log-file-name*, *mirror-file-name*, *iq-file-name*, *message-file-name*, *temp-file-name*) are strings containing operating system file names. As literal strings, they must be enclosed in single quotes.

- In Windows, if you specify a path, any backslash characters (\) must be doubled if they are followed by an n or an x. This prevents them being interpreted as a newline character (\n) or as a hexadecimal number (\x), according to the rules for strings in SQL. It is safer to always double the backslash. For example:

```
CREATE DATABASE 'c:\\sybase\\mydb.db'
LOG ON 'e:\\logdrive\\mydb.log'
JCONNECT OFF
IQ PATH 'c:\\sybase\\mydb'
IQ SIZE 40
```

- If you specify no path, or a relative path:
 - The catalog store file (*db-name.db*) is created relative to the working directory of the server.
 - The IQ store, temporary store, and message log files are created in the same directory as, or relative to, the catalog store.

Relative path names are recommended.

Warning! The database file, temporary dbspace, and transaction log file must be located on the same physical machine as the database server. Do not place database files and transaction log files on a network drive. The transaction log should be on a separate device from its mirror, however.

On UNIX-like operating systems, you can create symbolic links, which are indirect pointers that contain the path name of the file to which they point. You can use symbolic links as relative path names. There are several advantages to creating a symbolic link for the database file name:

- Symbolic links to raw devices can have meaningful names, while the actual device name syntax can be obscure.
- A symbolic name might eliminate problems restoring a database file that was moved to a new directory since it was backed up.

To create a symbolic link, use the **ln -s** command. For example:

```
ln -s /disk1/company/iqdata/company.iq company_iq_store
```

Once you create this link, you can specify the symbolic link in commands like **CREATE DATABASE** or **RESTORE** instead of the fully qualified path name.

When you create a database or a dbspace, the path for every dbspace file must be unique. If your **CREATE DATABASE** command specifies the identical path and file name for these two stores, you receive an error.

You can create a unique path in any of these ways:

- Specify a different extension for each file (for example, *mydb.iq* and *mydb.iqtmp*)
- Specify a different file name (for example, *mydb.iq* and *mytmp.iq*)
- Specify a different path name (for example, */iqfiles/main/iq* and */iqfiles/temp/iq*) or different raw partitions
- Omit **TEMPORARY PATH** when you create the database. In this case, the temporary store is created in the same path as the catalog store, with the default name and extension *dbname.iqtmp*, where *dbname* is the database name.

Warning! To maintain database consistency on UNIX-like operating systems, you must specify file names that are links to different files. SAP Sybase IQ cannot detect the target where linked files point. Even if the file names in the command differ, make sure they do not point to the same operating system file.

Character strings inserted into tables are always stored in the case they are entered, regardless of whether the database is case-sensitive or not. If the string Value is inserted into a character data type column, the string is always stored in the database with an uppercase V and the remainder of the letters lowercase. **SELECT** statements return the string as Value. If the database is not case-sensitive, however, all comparisons make Value the same as value, VALUE, and so on. The SAP Sybase IQ server may return results in any combination of lowercase and uppercase, so you cannot expect case-sensitive results in a database that is case-insensitive (CASE IGNORE).

For example, given this table and data:

```
CREATE TABLE tb (id int NOT NULL,  
                 string VARCHAR(30) NOT NULL);  
INSERT INTO tb VALUES (1, 'ONE');  
SELECT * FROM tb WHERE string = 'oNe';
```

The result of the **SELECT** can be “oNe” (as specified in the WHERE clause) and not necessarily “ONE” (as stored in the database).

Similarly, the result of:

```
SELECT * FROM tb WHERE string = 'One';
```

can be “One” and the result of:

```
SELECT * FROM tb WHERE string = 'ONE';
```

can be “ONE”.

All databases are created with at least one user ID:

```
DBA
```

and password:

```
sql
```

In new databases, all passwords are case-sensitive, regardless of the case-sensitivity of the database. The user ID is unaffected by the CASE RESPECT setting.

When you start a database, its page size cannot be larger than the page size of the current server. The server page size is taken from the first set of databases started or is set on the server command line using the **-gp** command line option.

Command line length for any statement is limited to the catalog page size. The 4KB default is large enough in most cases; however, in a few cases, a larger PAGE SIZE value is needed to accommodate very long commands, such as **RESTORE** commands that reference numerous dbspaces. A larger page size might also be needed to execute queries involving large numbers of tables or views.

Because the default catalog page size is 4KB, this is a problem only when the connection is to a database such as `utility_db`, which has a page size of 1024. This restriction may cause **RESTORE** commands that reference numerous dbspaces to fail. To avoid the problem, make sure the length of SQL command lines is less than the catalog page size.

Alternatively, start the engine with **-gp 32768** to increase catalog page size.

Side effects:

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Adaptive Server Enterprise provides a **CREATE DATABASE** statement, but with different options.

Permissions

The permissions required to execute this statement are set using the **-gu** server command line option, as follows:

- **NONE** – No user can issue this statement.
- **DBA** – Requires the SERVER OPERATOR system privilege.
- **UTILITY_DB** – Only those users who can connect to the `utility_db` database can issue this statement.

The account under which the server is running must have write permissions on the directories where files are created.

See also

- *CREATE DBSPACE Statement* on page 117
- *DROP DATABASE Statement* on page 260

CREATE DBSPACE Statement

Creates a new dbspace and the associated dbfiles for the IQ main store, catalog store, or RLV store.

Syntax

Syntax 1 – Use for catalog store dbspaces only (SQL Anywhere (SA) dbspaces).

```
CREATE DBSPACE dbspace-name AS file-path CATALOG STORE
```

Syntax 2 – Use for IQ main store dbspaces.

```
CREATE DBSPACE dbspace-name USING file-specification
[ IQ STORE ] iq-dbspace-opts
```

Syntax 3 – Use for RLV dbspaces.

```
CREATE DBSPACE dbspace-name USING file-specification
IQ RLV STORE
```

file-specification:

```
{ single-path-spec | new-file-spec [, ...] }
```

single-path-spec:

```
'file-path' | iq-file-opts
```

new-file-spec:

```
FILE logical-file-name | 'file-path' iq-file-opts
```

iq-file-opts:

```
[ [ SIZE ] file-size ]
...[ KB | MB | GB | TB ] ]
[ RESERVE size
...[ KB | MB | GB | TB ] ]
```

iq-dbspace-opts:

```
[ STRIPING ] { ON | OFF } ] ...[ STRIPESIZEKB sizeKB ]
```

Parameters

- **new-file-spec** – creates a dbspace for the IQ main store. You can specify one or more dbfiles for the IQ main store. The dbfile name and physical file path are required for each file, and must be unique.
- **RESERVE** – specifies the size in kilobytes (KB), megabytes (MB), gigabytes (GB), or terabytes (TB) of space to reserve, so that the dbspace can be increased in size in the future. The *size* parameter can be any number greater than 0; megabytes is the default. You cannot change the reserve after the dbspace dbfile is created.

When RESERVE is specified, the database uses more space for internal (free list) structures. If reserve size is too large, the space needed for the internal structures can be larger than the specified size, which results in an error.

- **dbspace-name and dbfile-name** – internal names for dbspaces and dbfiles. A database can have as many as (32KB - 1) dbspaces, including the initial dbspaces created when you create the database. However, your operating system might limit the number of dbfiles per database.
- **file-path** – the actual operating system file name of the dbfile, with a preceding path where necessary. *file-path* without an explicit directory is created in the same directory as the catalog store of the database. Any relative directory is relative to the catalog store.
- **SIZE** – specifies the size, from 0 to 4 terabytes, of the operating system file specified in *file-path*. The default depends on the store type and block size. For the IQ main store, the default number of bytes equals 1000* the block size. You cannot specify the SIZE clause

for the catalog store. A SIZE value of 0 creates a dbspace of minimum size, which is 8MB for the IQ main store.

For raw partitions, do not explicitly specify SIZE. SAP Sybase IQ automatically sets this parameter to the maximum raw partition size, and returns an error if you attempt to specify another size.

- **STRIPESIZEKB** – specifies the number of kilobytes (KB) to write to each file before the disk striping algorithm moves to the next stripe for the specified dbspace.

If you do not specify striping or stripe size, the default values of the options DEFAULT_DISK_STRIPING and DEFAULT_KB_PER_STRIPE apply.

Examples

- **Example 1** – creates a dbspace called DspHist for the IQ main store with two dbfiles on a UNIX system. Each dbfile is 1GB in size and can grow 500MB:

```
CREATE DBSPACE DspHist USING FILE
FileHist1 '/History1/data/file1'
SIZE 1000 RESERVE 500,
FILE FileHist2 '/History1/data/file2'
SIZE 1000 RESERVE 500;
```

- **Example 2** – creates a second catalog dbspace called DspCat2:

```
CREATE DBSPACE DspCat2 AS
'catalog_file2'
CATALOG STORE;
```

- **Example 3** – creates an IQ main dbspace called EmpStore1 for the IQ store (three alternate syntax examples):

```
CREATE DBSPACE EmpStore1
USING FILE EmpStore1
'EmpStore1.IQ' SIZE 8 MB IQ STORE;
```

```
CREATE DBSPACE EmpStore1
USING FILE EmpStore1
'EmpStore1.IQ' 8 IQ STORE;
```

```
CREATE DBSPACE EmpStore1
USING FILE EmpStore1
'EmpStore1.IQ' 8;
```

- **Example 4** – creates a RLV store dbspace called d1:

```
CREATE DBSPACE d1
USING FILE f1
'f1.iq' SIZE 100 IQ RLV STORE;
```

Usage

CREATE DBSPACE creates a new dbspace for the IQ main store, catalog store, or RLV store. The dbspace you add can be on a different disk device than the initial dbspace, allowing you to create stores that are larger than one physical device.

Syntax 1 creates a dbspace for the catalog store, where both dbspace and dbfile have the same logical name. Each dbspace in the catalog store has a single file.

The dbspace name and dbfile names are always case-insensitive. The physical file paths have the case sensitivity of the operating system if the database is CASE RESPECT, and are case-insensitive if the database is CASE IGNORE.

You cannot create a dbspace for an IQ temporary store. A single temporary dbspace, `IQ_SYSTEM_TEMP`, is created when you create a new database or upgrade one that was created in a version earlier than SAP Sybase IQ 15.3. You can add additional files to the `IQ_SYSTEM_TEMP` dbspace using the **ALTER DBSPACE ADD FILE** syntax.

Note: Creating a RLV dbspace containing a minimum of one file is a prerequisite for RLV storage. Before enabling RLV storage on a simplex server, check that the RLV dbspace exists.

You can create a unique path in any of these ways:

- Specify a different extension for each file (for example, `mydb.iq`)
- Specify a different file name (for example, `mydb2.iq`)
- Specify a different path name (for example, `/iqfiles/main/iq`) or different raw partitions

Warning! On UNIX platforms, to maintain database consistency, specify file names that are links to different files. SAP Sybase IQ cannot detect the target where linked files point. Even if the file names in the command differ, make sure they do not point to the same operating system file.

Side effects:

- Automatic commit
- Automatic checkpoint.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the `MANAGE ANY DBSPACE` system privilege.

See also

- *CREATE DATABASE Statement* on page 107
- *DROP Statement* on page 255

CREATE DOMAIN Statement

Creates a user-defined data type in the database.

Syntax

```
CREATE { DOMAIN | DATATYPE } domain-name data-type
... [ NOT ] NULL ]
... [ DEFAULT ]
```

default-value:

```
special-value
| string
| global variable
| [ - ] number
| ( constant-expression )
| built-in-function( constant-expression )
| AUTOINCREMENT
| CURRENT DATABASE
| CURRENT REMOTE USER
| NULL
| TIMESTAMP
| LAST USER
```

special-value:

```
CURRENT
{ DATE
| TIME
| TIMESTAMP
| USER
| PUBLISHER }
| USER
```

Parameters

- **data-type** – built-in data type, with precision and scale

Examples

- **Example 1** – create a data type named `address`, which holds a 35-character string, and which may be NULL:

```
CREATE DOMAIN address CHAR( 35 ) NULL
```

Usage

User-defined data types are aliases for built-in data types, including precision and scale values, where applicable. They improve convenience and encourage consistency in the database.

Note: Use **CREATE DOMAIN**, rather than **CREATE DATATYPE**, as **CREATE DOMAIN** is the ANSI/ISO SQL3 term.

The user who creates a data type is automatically made the owner of that data type. No owner can be specified in the **CREATE DATATYPE** statement. The user-defined data type name must be unique, and all users can access the data type without using the owner as prefix.

User-defined data types are objects within the database. Their names must conform to the rules for identifiers. User-defined data type names are always case-insensitive, as are built-in data type names.

By default, user-defined data types allow NULLs unless the **allow_nulls_by_default** database option is set to OFF. In this case, new user-defined data types by default do not allow NULLs. The nullability of a column created on a user-defined data type depends on the setting of the definition of the user-defined data type, not on the setting of the **allow_nulls_by_default** option when the column is referenced. Any explicit setting of NULL or NOT NULL in the column definition overrides the user-defined data type setting.

The **CREATE DOMAIN** statement allows you to specify DEFAULT values on user-defined data types. The DEFAULT value specification is inherited by any column defined on the data type. Any DEFAULT value explicitly specified on the column overrides that specified for the data type.

The **CREATE DOMAIN** statement lets you incorporate a rule, called a CHECK condition, into the definition of a user-defined data type.

SAP Sybase IQ enforces CHECK constraints for base, global temporary, local temporary tables, and user-defined data types.

To drop the data type from the database, use the **DROP** statement. You must be either the owner of the data type or have the CREATE DATATYPE or CREATE ANY OBJECT system privilege in order to drop a user-defined data type.

Side effects:

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported by Adaptive Server Enterprise. Transact-SQL provides similar functionality using the **sp_addtype** system procedure and the **CREATE DEFAULT** and **CREATE RULE** statements.

Permissions

Requires one of:

- CREATE DATATYPE system privilege.
- CREATE ANY OBJECT system privilege.

See also

- *DROP Statement* on page 255

CREATE EVENT Statement

Defines an event and its associated handler for automating predefined actions. Also defines scheduled actions.

Syntax

```

CREATE EVENT event-name
  [ TYPE event-type
    [ WHERE trigger-condition [ AND trigger-condition ], ... ]
    | SCHEDULE schedule-spec, ... ]
...[ ENABLE | DISABLE ]
...[ AT { CONSOLIDATED | REMOTE | ALL } ]
...[ HANDLER
  BEGIN
  ...
  END ]

```

event-type:

```

BackupEnd
| "Connect"
| ConnectFailed
| DatabaseStart
| DBDiskSpace
| "Disconnect"
| GlobalAutoincrement
| GrowDB
| GrowLog
| GrowTemp
| IQMainDBSpaceFree
| IQTempDBSpaceFree
| LogDiskSpace
| "RAISERROR"
| ServerIdle
| TempDiskSpace

```

trigger-condition:

```

event_condition ( condition-name )
{ =

```

```
| <
| >
| !=
| <=
| >= } value
```

schedule-spec:

```
[ schedule-name ]
{ START TIME start-time | BETWEEN start-time AND end-time }
[ EVERY period { HOURS | MINUTES | SECONDS } ]
[ ON { ( day-of-week, ... ) | ( day-of-month, ... ) } ]
[ START DATE start-date ]
```

Parameters

- **event-name** – an event has a creator, which is the user creating the event, and the event handler executes with the permissions of that creator. This is the same as stored procedure execution. You cannot create events owned by other users. You can list event names by querying the system table SYSEVENT. For example:

```
SELECT event_id, event_name FROM SYS.SYSEVENT
```

- **event-type** – one of a set of system-defined event types. The event types are case-insensitive. To specify the conditions under which this *event-type* triggers the event, use the WHERE clause.
 - **DiskSpace** – if the database contains an event handler for one of the DiskSpace types, the database server checks the available space on each device associated with the relevant file every 30 seconds.
In the event the database has more than one dbspace, on separate drives, DBDiskSpace checks each drive and acts depending on the lowest available space.
 - **LogDiskSpace** – checks the location of the transaction log and any mirrored transaction log, and reports based on the least available space.
 - **Globalautoincrement** – fires when the GLOBAL AUTOINCREMENT default value for a table is within one percent of the end of its range. A typical action for the handler could be to request a new value for the GLOBAL_DATABASE_ID clause.
You can use the **EVENT_CONDITION** function with RemainingValues as an argument for this event type.
 - **ServerIdle** – if the database contains an event handler for the ServerIdle type, the server checks for server activity every 30 seconds.
- **WHERE clause** – the trigger condition determines the condition under which an event is fired. For example, to take an action when the disk containing the transaction log becomes more than 80% full, use this triggering condition:

```
...
WHERE event_condition( 'LogDiskSpacePercentFree' ) < 20
...
```

The argument to the `EVENT_CONDITION` function must be valid for the event type. You can use multiple `AND` conditions to make up the `WHERE` clause, but you cannot use `OR` conditions or other conditions.

- **SCHEDULE** – specifies when scheduled actions are to take place. The sequence of times acts as a set of triggering conditions for the associated actions defined in the event handler. You can create more than one schedule for a given event and its associated handler. This permits complex schedules to be implemented. While it is compulsory to provide a schedule name when there is more than one schedule, it is optional if you provide only a single schedule.

You can list schedule names by querying the system table `SYSSCHEDULE`. For example:

```
SELECT event_id, sched_name FROM SYS.SYSSCHEDULE
```

Each event has a unique event ID. Use the `event_id` columns of `SYSEVENT` and `SYSSCHEDULE` to match the event to the associated schedule.

When a nonrecurring scheduled event has passed, its schedule is deleted, but the event handler is not deleted.

Scheduled event times are calculated when the schedules are created, and again when the event handler completes execution. The next event time is computed by inspecting the schedule or schedules for the event, and finding the next schedule time that is in the future. If an event handler is instructed to run every hour between 9:00 and 5:00, and it takes 65 minutes to execute, it runs at 9:00, 11:00, 1:00, 3:00, and 5:00. If you want execution to overlap, you must create more than one event.

The subclasses of a schedule definition are as follows:

- **START DATE** – the date on which scheduled events are to start occurring. The default is the current date.
- **START TIME** – the first scheduled time for each day on which the event is scheduled. If a **START DATE** is specified, the **START TIME** refers to that date. If no **START DATE** is specified, the **START TIME** is on the current day (unless the time has passed) and each subsequent day.
- **BETWEEN ... AND** – a range of times during the day outside of which no scheduled times occur. If a **START DATE** is specified, the scheduled times do not occur until that date.
- **EVERY** – an interval between successive scheduled events. Scheduled events occur only after the **START TIME** for the day, or in the range specified by **BETWEEN ... AND**.
- **ON** – a list of days on which the scheduled events occur. The default is every day. These can be specified as days of the week or days of the month.

Days of the week are Monday, Tuesday, and so on. The abbreviated forms of the day, such as Mon, Tue, and so on, may also be used. The database server recognizes both full-length and abbreviated day names in any of the languages supported by SAP Sybase IQ.

SQL Statements

Days of the month are integers from 0 to 31. A value of 0 represents the last day of any month.

Each time a scheduled event handler is completed, the next scheduled time and date is calculated.

- If the **EVERY** clause is used, find whether the next scheduled time falls on the current day, and is before the end of the **BETWEEN ...AND** range. If so, that is the next scheduled time.
- If the next scheduled time does not fall on the current day, find the next date on which the event is to be executed.
- Find the **START TIME** for that date, or the beginning of the **BETWEEN ... AND** range.
- **ENABLE | DISABLE** – by default, event handlers are enabled. When **DISABLE** is specified, the event handler does not execute even when the scheduled time or triggering condition occurs. A **TRIGGER EVENT** statement does not cause a disabled event handler to be executed
- **AT** – to execute events at remote or consolidated databases in a **SQL Remote** setup, use this clause to restrict the databases at which the event is handled. By default, all databases execute the event.
- **HANDLER** – each event has one handler. Like the body of a stored procedure, the handler is a compound statement. There are some differences, though: you can use an **EXCEPTION** clause within the compound statement to handle errors, but not the **ON EXCEPTION RESUME** clause provided within stored procedures.

Examples

- **Example 1** – instructs the database server to carry out an automatic incremental backup daily at 1 a.m.:

```
CREATE EVENT IncrementalBackup
SCHEDULE
START TIME '1:00AM' EVERY 24 HOURS
HANDLER
BEGIN
    BACKUP DATABASE INCREMENTAL
    TO 'backups/daily.incr'
END
```

- **Example 2** – instructs the database server to call the system stored procedure **sp_iqspaceused** every 10 minutes, then store in a table the returned current date and time, the current number of connections to the database, and current information about the use of main and temporary IQ store:

```
CREATE TABLE mysummary(dt DATETIME,
    users INT, mainKB UNSIGNED BIGINT,
    mainPC UNSIGNED INT,
```



```

tempKB UNSIGNED BIGINT,
tempPC UNSIGNED INT) ;

CREATE EVENT mysummary
  SCHEDULE sched_mysummary
    START TIME '00:01 AM' EVERY 10 MINUTES
  HANDLER
  BEGIN
    DECLARE mt UNSIGNED BIGINT;
    DECLARE mu UNSIGNED BIGINT;
    DECLARE tt UNSIGNED BIGINT;
    DECLARE tu UNSIGNED BIGINT;
    DECLARE conncount UNSIGNED INT;

    SET conncount = DB_PROPERTY('ConnCount');
    CALL SP_IQSPACEUSED(mt,mu,tt,tu);

    INSERT INTO mysummary VALUES( NOW(),
conncount, mu, (mu*100)/mt, tu,
(tu*100)/tt );
  END;

```

- **Example 3** – posts a message to the server log when free disk space on the device containing the transaction log file falls below 30 percent, but execute the handler no more than once every 300 seconds.

```

CREATE EVENT LowTxnLogDiskSpace
  TYPE DBDiskSpace
  WHERE event_condition( 'DBFreePercent' ) < 30
  AND event_condition( 'Interval' ) >= 300
  HANDLER
  BEGIN
  message 'Disk space for Transaction Log is low.';
  END;

```

Usage

Events can be used in two main ways:

- **Scheduling actions** – the database server carries out a set of actions on a schedule of times. You can use this capability to schedule backups, validity checks, queries to fill up reporting tables, and so on.
- **Event handling actions** – the database server carries out a set of actions when a predefined event occurs. The events that can be handled include disk space restrictions (when a disk fills beyond a specified percentage), when the server is idle, and so on.

An event definition includes two distinct pieces. The trigger condition can be an occurrence, such as a disk filling up beyond a defined threshold. A schedule is a set of times, each of which acts as a trigger condition. When a trigger condition is satisfied, the event handler executes. The event handler includes one or more actions specified inside a compound statement (**BEGIN... END**).

If no trigger condition or schedule specification is supplied, only an explicit **TRIGGER EVENT** statement can trigger the event. During development, you might want to develop and test event

SQL Statements

handlers using **TRIGGER EVENT** and add the schedule or WHERE clause once testing is complete.

Event errors are logged to the database server console.

When event handlers are triggered, the server makes context information, such as the connection ID that caused the event to be triggered, available to the event handler using the **EVENT_PARAMETER** function.

Note: Although statements that return result sets are disallowed in events, you can allow an event to call a stored procedure and insert the procedure results into a temporary table.

Side Effects:

- Automatic commit.
- The actions of an event handler are committed if no error is detected during execution, and rolled back if errors are detected.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires one of:

- **MANAGE ANY EVENT** system privilege.
- **CREATE ANY OBJECT** system privilege.

Event handlers execute on a separate connection, with the privileges of the event owner. To execute with privileges other than **MANAGE ANY EVENT** system privilege, you can call a procedure from within the event handler: the procedure executes with the permissions of its owner. The separate connection does not count towards the ten-connection limit of the personal database server.

See also

- *ALTER EVENT Statement* on page 14
- *BEGIN ... END Statement* on page 84
- *COMMENT Statement* on page 96
- *DROP Statement* on page 255
- *TRIGGER EVENT Statement* on page 443

CREATE EXISTING TABLE Statement

Creates a new proxy table that represents an existing table on a remote server.

Syntax

```
CREATE EXISTING TABLE [owner.]table_name
  [ ( column-definition, ... ) ]
  AT 'location-string'
```

```
column-definition:
  column-name data-type [ NOT NULL ]
```

```
location-string:
  remote-server-name.[db-name].[owner].object-name | remote-server-
name;[db-name];[owner];object-name
```

Parameters

- **column-definition** – if you do not specify column definitions, SAP Sybase IQ derives the column list from the metadata it obtains from the remote table. If you do specify column definitions, SAP Sybase IQ verifies them. When SAP Sybase IQ checks column names, data types, lengths, and null properties:
 - Column names must match identically (although case is ignored).
 - Data types in **CREATE EXISTING TABLE** must match or be convertible to the data types of the column on the remote location. For example, a local column data type is defined as NUMERIC, whereas the remote column data type is MONEY. You may encounter some errors, if you select from a table in which the data types do not match or other inconsistencies exist.
 - Each column's NULL property is checked. If the local column's NULL property is not identical to the remote column's NULL property, a warning message is issued, but the statement is not aborted.
 - Each column's length is checked. If the lengths of CHAR, VARCHAR, BINARY, DECIMAL, and NUMERIC columns do not match, a warning message is issued, but the command is not aborted. You might choose to include only a subset of the actual remote column list in your **CREATE EXISTING** statement.
- **AT** – specifies the location of the remote object. The AT clause supports the semicolon (;) as a delimiter. If a semicolon is present anywhere in the location string, the semicolon is the field delimiter. If no semicolon is present, a period is the field delimiter. This allows you to use file names and extensions in the database and owner fields. Semicolon field delimiters are used primarily with server classes that are not currently supported; however, you can also use them where a period would also work as a field delimiter.

For example, this statement maps the table `proxy_a1` to the SQL Anywhere database `mydb` on the remote server `myasa`:

```
CREATE EXISTING TABLE
proxy_a1
AT 'myasa;mydb;;a1'
```

Examples

- **Example 1** – create a proxy table named `nation` for the `nation` table at the remote server `server_a`:

```
CREATE EXISTING TABLE nation
( n_nationkey int,
  n_name char(25),
  n_regionkey int,
  n_comment char(152))
AT 'server_a.dbl.joe.nation'
```

- **Example 2** – create a proxy table named `blurbs` for the `blurbs` table at the remote server `server_a`. SAP Sybase IQ derives the column list from the metadata it obtains from the remote table:

```
CREATE EXISTING TABLE blurbs
AT 'server_a.dbl.joe.blurbs'
```

- **Example 3** – create a proxy table named `rda_employee` for the `Employees` table at the SAP Sybase IQ remote server `remote_iqdemo_srv`:

```
CREATE EXISTING TABLE rda_employee
AT 'remote_iqdemo_srv..dba.Employees'
```

Usage

CREATE EXISTING TABLE is a variant of the **CREATE TABLE** statement. The **EXISTING** keyword is used with **CREATE TABLE** to specify that a table already exists remotely, and that its metadata is to be imported into SAP Sybase IQ. This establishes the remote table as a visible entity to its users. SAP Sybase IQ verifies that the table exists at the external location before it creates the table.

Tables used as proxy tables cannot have names longer than 30 characters.

If the object does not exist (either as a host data file or remote server object), the statement is rejected with an error message.

Index information from the host data file or remote server table is extracted and used to create rows for the system table `sysindexes`. This defines indexes and keys in server terms and enables the query optimizer to consider any indexes that might exist on this table.

Referential constraints are passed to the remote location when appropriate.

In a simplex environment, you cannot create a proxy table that refers to a remote table on the same node. In a multiplex environment, you cannot create a proxy table that refers to the remote table defined within the multiplex.

For example, in a simplex environment, if you try to create proxy table `proxy_e`, which refers to base table `Employees` defined on the same node, the **CREATE EXISTING TABLE** statement is rejected with an error message. In a multiplex environment, the **CREATE EXISTING TABLE** statement is rejected if you create proxy table `proxy_e` from any node (coordinator or secondary) that refers to remote table `Employees` defined within a multiplex.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

For table to be owned by self – Requires one of:

- CREATE ANY TABLE system privilege.
- CREATE ANY OBJECT system privilege.

For table to be owned by any user – Requires the CREATE ANY TABLE system privilege.

See also

- *CREATE TABLE Statement* on page 205

CREATE EXTERNLOGIN Statement

Assigns an alternate login name and password to be used when communicating with a remote server.

Syntax

```
CREATE EXTERNLOGIN login-name
  TO remote-server
  REMOTE LOGIN remote-user
  [ IDENTIFIED BY remote-password ]
```

Parameters

- **login-name** – specifies the local user login name. When using integrated logins, the *login-name* is the database user to which the Windows user ID is mapped.
- **TO** – specifies the name of the remote server.

SQL Statements

- **REMOTE LOGIN** – specifies the user account on *remote-server* for the local user *login-name*.
- **IDENTIFIED BY** – specifies that *remote-password* is the password for *remote-user*. If you omit the **IDENTIFIED BY** clause, the password is sent to the remote server as NULL. If you specify **IDENTIFIED BY** " " (an empty string), the password sent is the empty string.

Examples

- **Example 1** – maps the local user named DBA to the user *sa* with password 4TKNOX when connecting to the server *sybase1*:

```
CREATE EXTERNLOGIN dba
TO sybase1
REMOTE LOGIN sa
IDENTIFIED BY 4TKNOX
```

Usage

Changes made by **CREATE EXTERNLOGIN** do not take effect until the next connection to the remote server.

By default, SAP Sybase IQ uses the names and passwords of its clients whenever it connects to a remote server on behalf of those clients. **CREATE EXTERNLOGIN** assigns an alternate login name and password to be used when communicating with a remote server. It stores the password internally in encrypted form.

The *remote_server* must be known to the local server by an entry in the *ISYSSERVER* system table. For more information, see the *CREATE SERVER Statement*.

Creating a remote login with the **CREATE EXTERNLOGIN** statement and defining a remote server with a **CREATE SERVER** statement sets up an external login and password for the *INSERT...LOCATION* such that any user can use the login and password in any context. This avoids possible errors due to inaccessibility of the login or password, and is the recommended way to connect to a remote server.

Note: If you rely on the user ID and password of the current connection, and a user changes the password, you must stop and restart the server before the new password takes effect on the remote server. Remote logins created with **CREATE EXTERNLOGIN** are unaffected by changes to the password for the default user ID.

Sites with automatic password expiration should plan for periodic updates of passwords for external logins.

CREATE EXTERNLOGIN cannot be used from within a transaction.

The *remote-user* and *remote-password* combination must be valid on *remote-server*.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *DROP EXTERNLOGIN Statement* on page 261
- *INSERT Statement* on page 327
- *CREATE SERVER Statement* on page 192

CREATE FUNCTION Statement

Creates a user-defined function in the database. A function can be created for another user by specifying an owner name. Subject to permissions, a user-defined function can be used in exactly the same way as other non-aggregate functions.

Syntax

Syntax 1

```
CREATE [ OR REPLACE ] [ TEMPORARY ] FUNCTION [ owner. ] function-name
( [ parameter, ... ] )
[ SQL SECURITY { INVOKER | DEFINER } ]
RETURNS data-type ON EXCEPTION RESUME
| [ NOT ] DETERMINISTIC
{ compound-statement | AS tsql-compound-statement
| EXTERNAL NAME library-call
| EXTERNAL NAME java-call LANGUAGE JAVA }
```

parameter:

```
IN parameter-name data-type [ DEFAULT expression ]
```

routine-characteristics:

```
ON EXCEPTION RESUME | [ NOT ] DETERMINISTIC
```

tsql-compound-statement:

```
sql-statement
sql-statement ...
```

library-call:

```
'[ operating-system : ] function-name @ library; ...'
```

operating-system:

```

UNIX

java-call:
    '[ package-name.]class-name.method-name method-signature'

method-signature:
    ( [ field-descriptor, ...] ) return-descriptor

field-descriptor and return-descriptor:
    Z | B | S | I | J | F | D | C | V | [descriptor | L class-name];

```

Syntax 2

```

CREATE FUNCTION [ owner.] function-name ( [ parameter, ... ] )
    RETURNS data-type
    URL url-string
    [ HEADER header-string ]
    [ SOAPHEADER soap-header-string ]
    [ TYPE { 'HTTP[:{ GET | POST } ] ' | 'SOAP[:{ RPC | DOC } ] ' } ]
    [ NAMESPACE namespace-string ]
    [ CERTIFICATE certificate-string ]
    [ CLIENTPORT clientport-string ]
    [ PROXY proxy-string ]

url-string:
    ' { HTTP | HTTPS | HTTPS_FIPS } ://[user:password@]hostname[:port] [/path ] '

parameter:
    IN parameter-name data-type [ DEFAULT expression ]

```

Parameters

- **CREATE [OR REPLACE]** – parameter names must conform to the rules for database identifiers. They must have a valid SQL data type and be prefixed by the keyword **IN**, signifying that the argument is an expression that provides a value to the function.

The **CREATE** clause creates a new function, while the **OR REPLACE** clause replaces an existing function with the same name. When a function is replaced, the definition of the function is changed but the existing permissions are preserved. You cannot use the **OR REPLACE** clause with temporary functions.

- **TEMPORARY** – the function is visible only by the connection that created it, and that it is automatically dropped when the connection is dropped. Temporary functions can also be explicitly dropped. You cannot perform **ALTER**, **GRANT**, or **REVOKE** operations on them, and unlike other functions, temporary functions are not recorded in the catalog or transaction log.

Temporary functions execute with the permissions of their creator (current user), and can only be owned by their creator. Therefore, do not specify owner when creating a temporary function. They can be created and dropped when connected to a read-only database.

- **SQL SECURITY** – defines whether the function is executed as the INVOKER, the user who is calling the function, or as the DEFINER, the user who owns the function. The default is DEFINER.

When INVOKER is specified, more memory is used because annotation must be done for each user that calls the procedure. Also, name resolution is done as the invoker as well. Therefore, take care to qualify all object names (tables, procedures, and so on) with their appropriate owner.

- **data-type** – LONG BINARY and LONG VARCHAR are not permitted as return-value data types.
- **compound-statement** – a set of SQL statements bracketed by **BEGIN** and **END**, and separated by semicolons. See *BEGIN ... END Statement*.
- **tsql-compound-statement** – a batch of Transact-SQL statements.
- **external-name** – a wrapper around a call to a function in an external library and can have no other clauses following the RETURNS clause. The library name may include the file extension, which is typically `.dll` on Windows and `.so` on UNIX. In the absence of the extension, the software appends the platform-specific default file extension for libraries.

The external-name clause is not supported for temporary functions.

- **LANGUAGE JAVA** – a wrapper around a Java method. For information on calling Java procedures, see *CREATE PROCEDURE Statement*.
- **ON EXCEPTION RESUME** – uses Transact-SQL-like error handling. See *CREATE PROCEDURE Statement*.
- **[NOT] DETERMINISTIC** – function is re-evaluated each time it is called in a query. The results of functions not specified in this manner may be cached for better performance, and re-used each time the function is called with the same parameters during query evaluation.

Functions that have side effects, such as modifying the underlying data, should be declared as NOT DETERMINISTIC. For example, a function that generates primary key values and is used in an **INSERT ... SELECT** statement should be declared NOT DETERMINISTIC:

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
NOT DETERMINISTIC
BEGIN
    DECLARE keyval INTEGER;
    UPDATE counter SET x = x + increment;
    SELECT counter.x INTO keyval FROM counter;
    RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table
```

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
```

```
NOT DETERMINISTIC
BEGIN
    DECLARE keyval INTEGER;
    UPDATE counter SET x = x + increment;
    SELECT counter.x INTO keyval FROM counter;
    RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table
```

Functions may be declared as DETERMINISTIC if they always return the same value for given input parameters. All user-defined functions are treated as deterministic unless they are declared NOT DETERMINISTIC. Deterministic functions return a consistent result for the same parameters and are free of side effects. That is, the database server assumes that two successive calls to the same function with the same parameters will return the same result without unwanted side-effects on the semantics of the query.

- **URL** – for use only when defining an HTTP or SOAP web services client function. Specifies the URL of the web service. The optional user name and password parameters provide a means of supplying the credentials needed for HTTP basic authentication. HTTP basic authentication base-64 encodes the user and password information and passes it in the “Authentication” header of the HTTP request.

For web service client functions, the return type of SOAP and HTTP functions must one of the character data types, such as VARCHAR. The value returned is the body of the HTTP response. No HTTP header information is included. If more information is required, such as status information, use a procedure instead of a function.

Parameter values are passed as part of the request. The syntax used depends on the type of request. For HTTP:GET, the parameters are passed as part of the URL; for HTTP:POST requests, the values are placed in the body of the request. Parameters to SOAP requests are always bundled in the request body.

- **HEADER** – when creating HTTP web service client functions, use this clause to add or modify HTTP request header entries. Only printable ASCII characters can be specified for HTTP headers, and they are case-insensitive. For more information about how to use this clause, see the HEADER clause of the *CREATE PROCEDURE Statement*.
- **SOAPHEADER** – when declaring a SOAP Web service as a function, use this clause to specify one or more SOAP request header entries. A SOAP header can be declared as a static constant, or can be dynamically set using the parameter substitution mechanism (declaring IN, OUT, or INOUT parameters for hd1, hd2, and so on). A web service function can define one or more IN mode substitution parameters, but cannot define an INOUT or OUT substitution parameter.
- **TYPE** – specifies the format used when making the web service request. If SOAP is specified or no type clause is included, the default type SOAP:RPC is used. HTTP implies

HTTP:POST. Since SOAP requests are always sent as XML documents, HTTP:POST is always used to send SOAP requests.

- **NAMESPACE** – applies to SOAP client functions only and identifies the method namespace usually required for both SOAP:RPC and SOAP:DOC requests. The SOAP server handling the request uses this namespace to interpret the names of the entities in the SOAP request message body. The namespace can be obtained from the WSDL description of the SOAP service available from the web service server. The default value is the procedure's URL, up to but not including the optional path component.
- **CERTIFICATE** – to make a secure (HTTPS) request, a client must have access to the certificate used by the HTTPS server. The necessary information is specified in a string of semicolon-separated key/value pairs. The certificate can be placed in a file and the name of the file provided using the file key, or the whole certificate can be placed in a string, but not both. These keys are available:

Key	Abbreviation	Description
file		File name of certificate
certificate	cert	The certificate
company	co	Company specified in the certificate
unit		Company unit specified in the certificate
name		Common name specified in the certificate

Certificates are required only for requests that are either directed to an HTTPS server or can be redirected from an insecure to a secure server.

- **CLIENTPORT** – identifies the port number on which the HTTP client procedure communicates using TCP/IP. It is provided for and recommended only for connections across firewalls, as firewalls filter according to the TCP/UDP port. You can specify a single port number, ranges of port numbers, or a combination of both; for example, CLIENTPORT '85,90-97'.
- **PROXY** – specifies the URI of a proxy server. For use when the client must access the network through a proxy. Indicates that the procedure is to connect to the proxy server and send the request to the web service through it.

Examples

- **Example 1** – concatenates a `firstname` string and a `lastname` string:

```
CREATE FUNCTION fullname (
    firstname CHAR(30),
    lastname CHAR(30) )
RETURNS CHAR(61)
BEGIN
    DECLARE name CHAR(61);
```

SQL Statements

```
SET name = firstname || ' ' || lastname;  
RETURN (name);  
END
```

This example illustrates the use of the **fullname** function.

- Return a full name from two supplied strings:

```
SELECT fullname ('joe','smith')
```

fullname('joe', 'smith')
joe smith

- List the names of all employees:

```
SELECT fullname (givenname, surname)  
FROM Employees
```

fullname (givenname, surname)
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
Robert Breault
...

- **Example 2** – uses Transact-SQL syntax:

```
CREATE FUNCTION DoubleIt ( @Input INT )  
RETURNS INT  
AS  
DECLARE @Result INT  
SELECT @Result = @Input * 2  
RETURN @Result
```

The statement `SELECT DoubleIt(5)` returns a value of 10.

- **Example 3** – creates an external function written in Java:

```
CREATE FUNCTION dba.encrypt( IN name char(254) )  
RETURNS VARCHAR  
EXTERNAL NAME  
'Scramble.encrypt (Ljava/lang/String;)Ljava/lang/String;'  
LANGUAGE JAVA
```

Usage

To modify a user-defined function, or to hide the contents of a function by scrambling its definition, use the **ALTER FUNCTION** statement.

When functions are executed, not all parameters need to be specified. If a default value is provided in the **CREATE FUNCTION** statement, missing parameters are assigned the default values. If an argument is not provided by the caller and no default is set, an error is given.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

For function to be owned by self – Requires the CREATE PROCEDURE system privilege.

For function to be owned by any user – Requires one of:

- CREATE ANY PROCEDURE system privilege.
- CREATE ANY OBJECT system privilege.

To create a function containing an external reference, regardless of whether or not they are the owner of the function, also requires the CREATE EXTERNAL REFERENCE system privilege.

See also

- *ALTER FUNCTION Statement* on page 16
- *BEGIN ... END Statement* on page 84
- *CREATE PROCEDURE Statement* on page 165
- *DROP Statement* on page 255
- *RETURN Statement* on page 396

CREATE FUNCTION Statement (Java UDF)

Creates a new external Java table UDF function in the database.

Syntax

```
CREATE [ OR REPLACE | TEMPORARY ] FUNCTION [ owner.]function-name
  ( [ parameter, ... ] )
  [ SQL SECURITY { INVOKER | DEFINER } ]
  RETURNS data-type
  ON EXCEPTION RESUME
  | [ NOT ] DETERMINISTIC
  { compound-statement | AS tsql-compound-statement
  | EXTERNAL NAME 'java-call' LANGUAGE JAVA [ ALLOW | DISALLOW SERVER SIDE
  REQUESTS ] environment-name}
```

```
parameter:  
    IN parameter-name data-type [ DEFAULT expression ]  
  
tsql-compound-statement:  
    sql-statement  
    sql-statement ...  
  
java-call:  
    '[ package-name.]class-name.method-name method-signature'  
  
method-signature:  
    ( [ field-descriptor, ...] ) return-descriptor  
  
field-descriptor and return-descriptor:  
    Z | B | S | I | J | F | D | C | V | [descriptor | L class-name;
```

Parameters

- **CREATE [OR REPLACE]** – parameter names must conform to the rules for database identifiers. They must have a valid SQL data type and be prefixed by the keyword **IN**, signifying that the argument is an expression that provides a value to the function.

The **CREATE** clause creates a new function, while the **OR REPLACE** clause replaces an existing function with the same name. When a function is replaced, the definition of the function is changed but the existing permissions are preserved. You cannot use the **OR REPLACE** clause with temporary functions.

- **TEMPORARY** – the function is visible only by the connection that created it, and that it is automatically dropped when the connection is dropped. Temporary functions can also be explicitly dropped. You cannot perform **ALTER**, **GRANT**, or **REVOKE** operations on them, and unlike other functions, temporary functions are not recorded in the catalog or transaction log.

Temporary functions execute with the permissions of their creator (current user), and can only be owned by their creator. Therefore, do not specify owner when creating a temporary function. They can be created and dropped when connected to a read-only database.

- **SQL SECURITY** – defines whether the function is executed as the **INVOKER**, the user who is calling the function, or as the **DEFINER**, the user who owns the function. The default is **DEFINER**.

When **INVOKER** is specified, more memory is used because annotation must be done for each user that calls the procedure. Also, name resolution is done as the invoker as well. Therefore, take care to qualify all object names (tables, procedures, and so on) with their appropriate owner.

- **data-type** – **LONG BINARY** and **LONG VARCHAR** are not permitted as return-value data types.

- **compound-statement** – a set of SQL statements bracketed by **BEGIN** and **END**, and separated by semicolons. See *BEGIN ... END Statement*.
- **tsql-compound-statement** – a batch of Transact-SQL statements.
- **[NOT] DETERMINISTIC** – function is re-evaluated each time it is called in a query. The results of functions not specified in this manner may be cached for better performance, and re-used each time the function is called with the same parameters during query evaluation.

Functions that have side effects, such as modifying the underlying data, should be declared as NOT DETERMINISTIC. For example, a function that generates primary key values and is used in an **INSERT ... SELECT** statement should be declared NOT DETERMINISTIC:

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
NOT DETERMINISTIC
BEGIN
    DECLARE keyval INTEGER;
    UPDATE counter SET x = x + increment;
    SELECT counter.x INTO keyval FROM counter;
    RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table
```

```
CREATE FUNCTION keygen( increment INTEGER )
RETURNS INTEGER
NOT DETERMINISTIC
BEGIN
    DECLARE keyval INTEGER;
    UPDATE counter SET x = x + increment;
    SELECT counter.x INTO keyval FROM counter;
    RETURN keyval
END
INSERT INTO new_table
SELECT keygen(1), ...
FROM old_table
```

Functions may be declared as DETERMINISTIC if they always return the same value for given input parameters. All user-defined functions are treated as deterministic unless they are declared NOT DETERMINISTIC. Deterministic functions return a consistent result for the same parameters and are free of side effects. That is, the database server assumes that two successive calls to the same function with the same parameters will return the same result without unwanted side-effects on the semantics of the query.

- **LANGUAGE JAVA** – a wrapper around a Java method. For information on calling Java procedures, see *CREATE PROCEDURE Statement*.
- **environment-name** – a wrapper around a Java method.

The **DISALLOW** clause is the default. The **ALLOW** clause indicates that server-side connections are allowed.

Note: Do not specify the **ALLOW** clause unless necessary. **ALLOW** slows down certain types of SAP Sybase IQ table joins. Do not use UDFs with both the **ALLOW** and **DISALLOW SERVER SIDE REQUESTS** clauses in the same query.

Examples

- **Example 1** – creates an external function written in Java:

```
CREATE FUNCTION dba.encrypt ( IN name char(254) )
RETURNS VARCHAR
EXTERNAL NAME
'Scramble.encrypt (Ljava/lang/String;)Ljava/lang/String;'
LANGUAGE JAVA
```

Usage

When functions are executed, not all parameters need to be specified. If a default value is provided in the **CREATE FUNCTION** statement, missing parameters are assigned the default values. If an argument is not provided by the caller and no default is set, an error is given.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

For function to be owned by self – Requires the **CREATE PROCEDURE** system privilege

For function to be owned by any user – Requires one of:

- **CREATE ANY PROCEDURES** system privilege.
- **CREATE ANY OBJECT** system privilege.

To create a function containing an external reference, regardless of whether or not they are the owner of the function, also requires the **CREATE EXTERNAL REFERENCE** system privilege.

CREATE INDEX Statement

Creates an index on a specified table, or pair of tables. Once an index is created, it is never referenced in a SQL statement again except to delete it using the **DROP INDEX** statement.

Syntax

```
CREATE [ UNIQUE ] [ index-type ] INDEX [ IF NOT EXISTS ] index-name
...ON [ owner.]table-name
... ( column-name [ , column-name ] ...)
```



```

...[ { IN | ON } dbspace-name ]
...[ NOTIFY integer ]
...[ DELIMITED BY `separators-string ` ]
...[ LIMIT maxwordsize-integer ]

```

index-type:

```
{ CMP | HG | HNG | LF | WD | DATE | TIME | DTTM }
```

Parameters

- **index-type** – for columns in SAP Sybase IQ tables, you can specify an index-type of HG (High_Group), HNG (High_Non_Group), LF (Low_Fast), WD (Word), DATE, TIME, or DTTM (Datetime). If you do not specify an index-type, an HG index is created by default.

To create an index on the relationship between two columns in an IQ main store table, you can specify an index-type of CMP (Compare). Columns must be of identical data type, precision and scale. For a CHAR, VARCHAR, BINARY or VARBINARY column, precision means that both columns have the same width.

For maximum query speed, the correct type of index for a column depends on:

- The number of unique values in the column
- How the column is going to be used in queries
- The amount of disk space available

You can specify multiple indexes on a column of an IQ main store table, but these must be of different index types. **CREATE INDEX** does not let you add a duplicate index type. SAP Sybase IQ chooses the fastest index available for the current query or portion of the query. However, each additional index type might significantly add to the space requirements of that table.

- **column-name** – specifies the name of the column to be indexed. A column name is an identifier preceded by an optional correlation name. (A correlation name is usually a table name. For more information on correlation names, see *FROM Clause*.) If a column name has characters other than letters, digits, and underscore, enclose it in quotation marks (“”).

If you omit the UNIQUE clause, you can specify only an HG index. Foreign keys require nonunique indexes and composite foreign keys require nonunique composite HG indexes. The multicolumn composite key for both unique and nonunique HG indexes has a maximum width of 5300 bytes. CHAR or VARCHAR data cannot be more than 255 bytes when it is part of a composite key or single-column HG, LF, HNG, DATE, TIME, or DTTM indexes.

- **UNIQUE** – ensures that no two rows in the table have identical values in all the columns in the index. Each index key must be unique or contain a NULL in at least one column. You can create unique HG indexes with more than one column, but you cannot create multicolumn indexes using other index types. You cannot specify UNIQUE with the CMP, HNG, WD, DATE, TIME, or DTTM index types.

SAP Sybase IQ allows the use of NULL in data values on a user created unique multicolumn HG index, if the column definition allows for NULL values and a constraint (primary key or unique) is not being enforced. See “Multicolumn indexes” in *Notes* for more information.

- **IF NOT EXISTS** – if the named object already exists, no changes are made and an error is not returned.
- **IN** – specifies index placement. If you omit the IN clause, the index is created in the dbspace where the table is created. An index is always placed in the same type of dbspace (IQ store or temporary store) as its table. When you load the index, the data is spread across any database files of that type with room available. SAP Sybase IQ ensures that any *dbspace-name* you specify is appropriate for the index. If you try to specify IQ_SYSTEM_MAIN or other main dbspaces for indexes on temporary tables, or vice versa, you receive an error. Dbspace names are always case-insensitive, regardless of the **CREATE DATABASE...CASE IGNORE** or **CASE RESPECT** specification.
- **DELIMITED BY** – specifies separators to use in parsing a column string into the words to be stored in the WD index of that column. If you omit this clause or specify the value as an empty string, SAP Sybase IQ uses the default set of separators. The default set of separators is designed for the default collation order (ISO-BINENG). It includes all 7-bit ASCII characters that are not 7-bit ASCII alphanumeric characters, except for the hyphen and the single quotation mark. The hyphen and the single quotation mark are part of words by default. There are 64 separators in the default separator set. For example, if the column value is this string:

```
The cat is on the mat
```

and the database was created with the CASE IGNORE setting using default separators, these words are stored in the WD index from this string:

```
cat is mat on the
```

If you specify multiple DELIMITED BY and LIMIT clauses, no error is returned, but only the last clause of each type is used.

- **separators-string** – must be a sequence of 0 or more characters in the collation order used when the database was created. Each character in the separators string is treated as a separator. If there are no characters in the separators string, the default set of separators is used. (Each separator must be a single character in the collation sequence being used.) There cannot be more than 256 characters (separators) in the separators string.

To specify tab as a delimiter, you can either type a <TAB> character within the separator string, or use the hexadecimal ASCII code of the tab character, \x09. “\t” specifies two separators, \ and the letter t. To specify newline as a delimiter, you can type a <RETURN> character or the hexadecimal ASCII code \x0a.

For example, the clause `DELIMITED BY ' ; ; . \ / t '` specifies these seven separators:
space ; ; . \ / t

Table 4. Tab and Newline as Delimiters

For these delimiters	Use this separators string in the DELIMITED BY clause
tab	' ' (type <TAB>)or '\x09'
newline	' ' (type <RETURN>) or '\x0a'

- **LIMIT** – can be used for the creation of the WD index only. Specifies the maximum word length that is permitted in the WD index. Longer words found during parsing causes an error. The default is 255 bytes. The minimum permitted value is 1 and the maximum permitted value is 255. If the maximum word length specified in the **CREATE INDEX** statement or determined by default exceeds the column width, the used maximum word length is silently reduced to the column width. Using a lower maximum permitted word length allows insertions, deletions, and updates to use less space and time. The empty word (two adjacent separators) is silently ignored. After a WD index is created, any insertions into its column are parsed using the separators and maximum word size determined at create time. These separators and maximum word size cannot be changed after the index is created.
- **NOTIFY** – gives notification messages after n records are successfully added for the index. The messages are sent to the standard output device. A message contains information about memory usage, database space, and how many buffers are in use. The default is 100,000 records. To turn off NOTIFY, set it to 0.
- –

Examples

- **Example 1** – creates a Compare index on the `projected_earnings` and `current_earnings` columns. These columns are decimal columns with identical precision and scale.

```
CREATE CMP INDEX proj_curr_cmp
ON sales_data
( projected_earnings, current_earnings )
```

- **Example 2** – creates a High_Group index on the ID column of the `SalesOrderItems` table. The data pages for this index are allocated from `dbspace Dsp5`.

```
CREATE HG INDEX id_hg
ON SalesOrderItems
( ID ) IN Dsp5
```

- **Example 3** – creates a High_Group index on the `SalesOrderItems` table for the `ProductID` column:

SQL Statements

```
CREATE HG INDEX item_prod_hg
ON Sales_OrderItems
( ProductID)
```

- **Example 4** – creates a **Low_Fast** index on the `SalesOrderItems` table for the same `ProductID` column without any notification messages:

```
CREATE LF INDEX item_prod
ON SalesOrderItems
( ProductID)
NOTIFY 0
```

- **Example 5** – creates a **WD** index on the `earnings_report` table. Specify that the delimiters of strings are space, colon, semicolon, and period. Limit the length of the strings to 25.

```
CREATE WD INDEX earnings_wd
ON earnings_report_table(varchar)
DELIMITED BY ' :;. '
LIMIT 25
```

- **Example 6** – creates a **DTTM** index on the `SalesOrders` table for the `OrderDate` column:

```
CREATE DTTM INDEX order_dttm
ON SalesOrders
( OrderDate )
```

Usage

- Index ownership—There is no way to specify the index owner in the **CREATE INDEX** statement. Indexes are automatically owned by the owner of the table on which they are defined. The index name must be unique for each owner.
- No indexes on views—Indexes cannot be created for views.
- Index name—The name of each index must be unique for a given table.
- Exclusive table use—**CREATE INDEX** is prevented whenever the statement affects a table currently being modified by another connection. However, queries are allowed on a table that is also adding an index.
- **CHAR** columns—After a **WD** index is created, any insertions into its column are parsed using the separators, and maximum word size cannot be changed after the index is created. For **CHAR** columns, specify a space as at least one of the separators or use the default separator set. SAP Sybase IQ automatically pads **CHAR** columns to the maximum column width. If your column contains blanks in addition to the character data, queries on **WD** indexed data might return misleading results. For example, column `CompanyName` contains two words delimited by a separator, but the second word is blank padded:

```
'Concord' 'Farms'
```

Suppose that a user entered this query:

```
SELECT COUNT(*) FROM Customers WHERE CompanyName contains ('Farms')
```

The parser determines that the string contains:

```
'Farms'
```

instead of:

```
'Farms'
```

and returns 0 instead of 1. You can avoid this problem by using `VARCHAR` instead of `CHAR` columns.

- Data types:
 - You cannot use **CREATE INDEX** to create an index on a column with `BIT` data.
 - Only the default index, **CMP** index, or **WD** index can be created on `CHAR` and `VARCHAR` data with more than 255 bytes.
 - Only the default and **WD** index types can be created on `LONG VARCHAR` data.
 - Only the default index, **CMP** index, and **TEXT** index types can be created on `BINARY` and `VARBINARY` data with more than 255 bytes.
 - An **HNG** index or a **CMP** index cannot be created on a column with `FLOAT`, `REAL`, or `DOUBLE` data.
 - A **TIME** index can be created only on a column having the data type `TIME`.
 - A **DATE** index can be created only on a column having the data type `DATE`.
 - A **DTTM** index can be created only on a column having the data type `DATETIME` or `TIMESTAMP`.
- Multicolumn indexes—You can create a unique or nonunique **HG** index with more than one column. SAP Sybase IQ implicitly creates a nonunique **HG** index on a set of columns that makes up a foreign key.

HG and **CMP** are the only types of indexes that can have multiple columns. You cannot create a unique **HNG** or **LF** index with more than one column, and you cannot create a **DATE**, **TIME**, or **DTTM** index with more than one column.

The maximum width of a multicolumn concatenated key is 5KB (5300 bytes). The number of columns allowed depends on how many columns can fit into 5KB. `CHAR` or `VARCHAR` data greater than 255 bytes are not allowed as part of a composite key in single-column **HG**, **LF**, **HNG**, **DATE**, **TIME**, or **DTTM** indexes.

An **INSERT** on a multicolumn index must include all columns of the index.

Queries with a single column in the **ORDER BY** clause run faster using multicolumn **HG** indexes. For example:

```
SELECT abs (x) from t1
ORDER BY x
```

In the above example, the **HG** index vertically projects `x` in sorted order.

To enhance query performance, use multicolumn **HG** indexes to run **ORDER BY** operations on more than one column (that can also include **ROWID**) in the **SELECT** or **ORDER BY** clause with these conditions:

- All projected columns, plus all ordering columns (except **ROWID**), exist within the index

SQL Statements

- The ordering keys match the leading **HG** columns, in order

If more than one multicolumn **HG** index satisfies these conditions, the index with the lowest distinct counts is used.

If a query has an **ORDER BY** clause, and the **ORDER BY** column list is a prefix of a multicolumn index where all columns referenced in the **SELECT** list are present in a multicolumn index, then the multicolumn index performs vertical projection; for example:

```
SELECT x, z, y FROM T
ORDER BY x, y
```

If expressions exist on base columns in the **SELECT** list, and all the columns referenced in all the expressions are present in the multicolumn index, then the query will use a multicolumn index; for example:

```
SELECT power(x,2), x+y, sin(z) FROM T
ORDER BY x, y
```

In addition to the two previous examples, if the **ROWID()** function is in the **SELECT** list expressions, multicolumn indexes will be used. For example:

```
SELECT rowid()+x, z FROM T
ORDER BY x, y, z
```

In addition to the three previous examples, if **ROWID()** is present at the end of an **ORDER BY** list, and if the columns of that list—except for **ROWID()**—use multicolumn indexes in the exact order, multicolumn indexes will be used for the query. For example:

```
SELECT z, y FROM T
ORDER BY x, y, z, ROWID()
```

SAP Sybase IQ allows the use of NULL in data values on a user created unique multicolumn **HG** index, if the column definition allows for NULL values and a constraint (primary key or unique) is not being enforced. The rules for this feature are as follows:

- A NULL is treated as an undefined value.
- Multiple rows with NULL values in a unique index column or columns are allowed.
 1. In a single column index, multiple rows with a NULL value in an index column are allowed.
 2. In a multicolumn index, multiple rows with a NULL value in index column or columns are allowed, as long as non-NULL values in the rest of the columns guarantee uniqueness in that index.
 3. In a multicolumn index, multiple rows with NULL values in all columns participating in the index are allowed.

These examples illustrate these rules. Given the table `table1`:

```
CREATE TABLE table1
(c1 INT NULL, c2 INT NULL, c3 INT NOT NULL);
```

Create a unique single column **HG** index on a column that allows NULLs:

```
CREATE UNIQUE HG INDEX c1_hg1 ON table1 (c1);
```

According to rule 1 above, you can insert a NULL value into an index column in multiple rows:

```
INSERT INTO table1 (c1, c2, c3) VALUES (NULL, 1, 1);
INSERT INTO table1 (c1, c2, c3) VALUES (NULL, 2, 2);
```

Create a unique multicolumn **HG** index on a columns that allows NULLs:

```
CREATE UNIQUE HG INDEX c1c2_hg2 ON table1 (c1, c2);
```

According to rule 2 above, you must guarantee uniqueness in the index. The following **INSERT** does not succeed, since the multicolumn index `c1c2_hg2` on row 1 and row 3 has the same value:

```
INSERT INTO table1 (c1, c2, c3) VALUES (NULL, 1, 3);
```

These **INSERT** operations are successful, however, according to rules 1 and 3:

```
INSERT INTO table1 (c1, c2, c3) VALUES (NULL, NULL, 3);
INSERT INTO table1 (c1, c2, c3) VALUES (NULL, NULL, 4);
```

Uniqueness is preserved in the multicolumn index.

This **UPDATE** operation is successful, as rule 3 allows multiple rows with NULL values in all columns in the multicolumn index:

```
UPDATE table1 SET c2=NULL WHERE c3=1
```

When a multicolumn **HG** index is governed by a unique constraint, a NULL value is not allowed in any column participating in the index.

- Parallel index creation—You can use the **BEGIN PARALLEL IQ ... END PARALLEL IQ** statement to group **CREATE INDEX** statements on multiple IQ main store tables, so that they execute as though they are a single DDL statement. See *BEGIN PARALLEL IQ ... END PARALLEL IQ Statement* for more information.

Warning! Using the **CREATE INDEX** command on a local temporary table containing uncommitted data fails and generates the error message `Local temporary table, <tablename>, must be committed in order to create an index.` Commit the data in the local temporary table before creating an index.

Side Effects

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Adaptive Server Enterprise has a more complex **CREATE INDEX** statement than SAP Sybase IQ. While the Adaptive Server Enterprise syntax is permitted in SAP Sybase IQ, some clauses and keywords are ignored. For the full syntax of the Adaptive Server Enterprise **CREATE INDEX** statement, see the *Adaptive Server Enterprise Reference Manual, Volume 2: Commands*.

Adaptive Server Enterprise indexes can be either *clustered* or *nonclustered*. A clustered index almost always retrieves data faster than a nonclustered index. Only one clustered index is permitted per table.

SQL Statements

SAP Sybase IQ does not support clustered indexes. The **CLUSTERED** and **NONCLUSTERED** keywords are allowed by SQL Anywhere, but are ignored by SAP Sybase IQ. If no *index-type* is specified, SAP Sybase IQ creates an HG index on the specified column(s).

SAP Sybase IQ does not permit the **DESC** keyword.

Index names must be unique on a given table for both SAP Sybase IQ and Adaptive Server Enterprise.

Permissions

Requires **CREATE** privilege on the dbspace where the index is being created. Also requires one of::

- **CREATE ANY INDEX** system privilege.
- **CREATE ANY OBJECT** system privilege.
- **REFERENCE** privilege on the underlying table of the index.
- You own the underlying table of the index.

See also

- *BEGIN PARALLEL IQ ... END PARALLEL IQ Statement* on page 87
- *DROP Statement* on page 255
- *INDEX_PREFERENCE Option* on page 557
- *FROM Clause* on page 295

CREATE LDAP SERVER Statement

Creates a new LDAP server configuration object for LDAP user authentication. Parameters defined during the creation of an LDAP server configuration object are stored in the **ISYSLDAPSERVER** (system view **SYSLDAPSERVER**) system table.

Syntax

```
CREATE LDAP SERVER ldapua-server-name
  [ ldapua-server-attrs ]
  [ WITH ACTIVATE ]

ldapua-server-attrs:
  SEARCH DN
    URL { 'URL_string' | NULL }
    | ACCESS ACCOUNT { 'DN_string' | NULL }
    | IDENTIFIED BY ( 'password' | NULL )
    | IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
  | AUTHENTICATION URL { 'URL_string' | NULL }
  | CONNECTION TIMEOUT timeout_value
  | CONNECTION RETRIES retry_value
```



```
| TLS { ON | OFF }
```

Parameters

- **URL** – identifies the host (by name or by IP address), port number, and the search to be performed for the DN lookup for a given user ID. This value is validated for correct LDAP URL syntax before it is stored in the `ISYSLDAPSERVER` system table. The maximum size for this string is 1024 bytes.
- **ACCESS ACCOUNT** – user created in the LDAP server for use by SAP Sybase IQ, not a user within SAP Sybase IQ. The distinguished name (DN) for this user is used to connect to the LDAP server. This user has permissions within the LDAP server to search for DNs by user ID in the locations specified by the `SEARCH DN URL`. The maximum size for this string is 1024 bytes.
- **IDENTIFIED BY** – provides the password associated with the `ACCESS ACCOUNT` user. The password is stored using symmetric encryption on disk. Use the value `NULL` to clear the password and set it to none. The maximum size of a clear text password is 255 bytes.
- **IDENTIFIED BY ENCRYPTED** – configures the password associated with the `ACCESS ACCOUNT` distinguished name in an encrypted format. The binary value is the encrypted password and is stored on disk as is. Use the value `NULL` to clear the password and set it to none. The maximum size of the binary is 289 bytes. The encrypted key should be a valid varbinary value. Do not enclose the encrypted key in quotation marks.
- **AUTHENTICATION URL** – identifies the host (by name or IP address) and the port number of the LDAP server to use for authentication of the user. This is the value defined for `URL_string` and is validated for correct LDAP URL syntax before it is stored in `ISYSLDAPSERVER` system table. The DN of the user obtained from a prior DN search and the user password bind a new connection to the authentication URL. A successful connection to the LDAP server is considered proof of the identity of the connecting user. The maximum size for this string is 1024 bytes.
- **CONNECTION TIMEOUT** – specifies the connection timeout from SAP Sybase IQ to the LDAP server for both DN searches and authentication. This value is in milliseconds, with a default value of 10 seconds.
- **CONNECTION RETRIES** – specifies the number of retries on connections from SAP Sybase IQ to the LDAP server for both DN searches and authentication. The valid range of values is 1– 60, with a default value of 3.
- **TLS** – defines whether the TLS or Secure LDAP protocol is used for connections to the LDAP server for both DN searches and authentication. When set to `ON`, the TLS protocol is used and the URL would begin with "ldap://". When set to `OFF` (or not specified), Secure LDAP protocol is used and the URL begins with "ldaps://". When using the TLS protocol, specify the database security option `TRUSTED_CERTIFICATES_FILE` with a file name

containing the certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server.

- **WITH ACTIVATE** – activates the LDAP server configuration object for immediate use upon creation. This permits the definition and activation of LDAP User Authentication in one statement. The LDAP server configuration object state changes to **READY** when **WITH ACTIVATE** is used.

Examples

- **Example 1** – sets the search parameters, the authentication URL, and sets a three second timeout, and activates the server so it can begin authenticating users. It connects to the LDAP server without TLS or SECURE LDAP protocols.

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'  
CREATE LDAP SERVER apps_primary  
SEARCH DN  
    URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'  
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'  
    IDENTIFIED BY 'Secret99Password'  
AUTHENTICATION URL 'ldap://my_LDAPserver:389/'  
CONNECTION TIMEOUT 3000  
WITH ACTIVATE
```

- **Example 2** – uses the same search parameters as example 1, but specifies “ldaps” so that a Secure LDAP connection is established with the LDAP server on host my_LDAPserver, port 636. Only LDAP clients using the Secure LDAP protocol may now connect on this port. The database security option TRUSTED_CERTIFICATE_FILE must be set with a file name containing the certificate of the certificate authority (CA) that signed the certificate used by the LDAP server at "ldaps://my_LDAPserver:636". During the handshake with the LDAP server, the certificate presented by the LDAP server is checked by the SAP Sybase IQ server (the LDAP client) to ensure that it is signed by one of the certificates listed in the file. This establishes trust by the client that the server is who it says it is. The ACCESS ACCOUNT and IDENTIFIED BY parameters establish trust by the LDAP server that the client is who it says it is.

Note: The TLS parameter must be OFF when Secure LDAP is used instead of TLS protocol.

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'  
SET OPTION PUBLIC.trusted_certificates_file = '/mycompany/shared/  
trusted.txt'  
CREATE LDAP SERVER secure_primary  
SEARCH DN  
    URL 'ldaps://my_LDAPserver:636/dc=MyCompany,dc=com??sub?  
cn=*'  
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'  
    IDENTIFIED BY 'Secret99Password'  
AUTHENTICATION URL 'ldaps://my_LDAPserver:636/'  
CONNECTION TIMEOUT 3000
```

```
TLS OFF
WITH ACTIVATE
```

- **Example 3** – establishes the TLS protocol on port 389. It also requires database security option TRUSTED_CERTIFICATE_FILE to be set with a file name and provides the same type of security as example 2. In this example, the TLS protocol is ON to facilitate wider support by LDAP server vendors.

Note: Check the requirements of all your LDAP servers when deciding how to configure Secure LDAP or TLS for an SAP Sybase IQ server.

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'
SET OPTION PUBLIC.trusted_certificates_file = '/mycompany/shared/
trusted.txt'
CREATE LDAP SERVER tls_primary
SEARCH DN
    URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'
    IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldap://my_LDAPserver:389/'
CONNECTION TIMEOUT 3000
TLS ON
WITH ACTIVATE
```

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE ANY LDAP SERVER system privilege.

CREATE LOGICAL SERVER Statement

Creates a user-defined logical server. This statement enforces consistent shared system temporary store settings across physical nodes shared by logical servers.

Syntax

```
CREATE LOGICAL SERVER logical-server-name [
    { ls-create-clause, ... } ] [ WITH STOP SERVER ]

ls-create-clause:
    { MEMBERSHIP ( { ls-member, ... } ) | POLICY ls-policy-name }

ls-member:
    FOR LOGICAL COORDINATOR | mpx-server-name
```

Parameters

- **logical-server-name** – any user-specified identifier except:
 - ALL
 - AUTO
 - COORDINATOR
 - DEFAULT
 - NONE
 - OPEN
 - SERVER
- **MEMBERSHIP** – to define a logical membership to the coordinator, include FOR LOGICAL COORDINATOR in the MEMBERSHIP clause.

When no members are specified during the creation of a logical server, the logical server is created empty.

Note: Implicit logical server membership definitions, such as those for OPEN and SERVER logical servers, are not stored at all.

The SYS . ISYSLOGICALMEMBER system table stores definitions for the logical server memberships.

Changing the **ALLOW_COORDINATOR_AS_MEMBER** option of the root logical server policy from ON to OFF does not affect the membership information stored in the catalog. Instead, it affects only the effective configuration of the logical server.

You can define a logical server membership to the current coordinator either by specifying the multiplex server name or by using the FOR LOGICAL COORDINATOR clause, even when **ALLOW_COORDINATOR_AS_MEMBER** option is set to OFF. Membership definition is stored in the catalog, but is inactive while that multiplex server acts as the coordinator.

The catalog stores the logical server and its membership definitions.

- **POLICY** – associates a logical server with a user-defined logical server policy. If no POLICY clause is specified, the logical server is associated with the root policy.

The SYS . ISYSIQLOGICALSERVER system table stores information about the logical server policy for a corresponding logical server.

- **ls-policy-name** – any user-specified identifier except ROOT.
- **WITH STOP SERVER** – automatically shuts down all servers in the logical server when the TEMP_DATA_IN_SHARED_TEMP option is changed directly or indirectly.

Applies to

Multiplex only.

Examples

- **Example 1** – creates a user-defined logical server *ls1* with three multiplex nodes as its members:

```
CREATE LOGICAL SERVER ls1 MEMBERSHIP ( n1, n2, n3 )
```

- **Example 2** – creates a user-defined logical server *ls1* with three member nodes, and defines the logical server policy name *lsp1*:

```
CREATE LOGICAL SERVER ls1 MEMBERSHIP ( w1_svr, w2_svr, r2_svr )
POLICY lsp1
```

- **Example 3** – creates servers as in Example 2, except that **WITH STOP SERVER** automatically shuts down all servers in the logical server when the **TEMP_DATA_IN_SHARED_TEMP** option is changed directly or indirectly:

```
CREATE LOGICAL SERVER ls1 MEMBERSHIP ( w1_svr, w2_svr, r2_svr )
POLICY lsp1 WITH STOP SERVER
```

- **Example 4** – creates a user-defined logical server *ls1* with logical server policy *lspolicy1* and no member nodes:

```
CREATE LOGICAL SERVER ls1 POLICY lspolicy1
```

- **Example 5** – where *n1* is the current coordinator, creates a logical server *ls2* with the named membership of multiplex nodes *n1* and *n3* and logical membership of the coordinator. Also sets the logical server policy of *ls2* to *lspolicy2*.

```
CREATE LOGICAL SERVER ls2 POLICY
MEMBERSHIP FOR LOGICAL COORDINATOR
lspolicy1, n1, n2, n3 POLICY lspolicy2
```

Permissions

Requires the **MANAGE MULTIPLEX** system privilege.

CREATE LOGIN POLICY Statement

Creates a login policy in the database.

Syntax

```
CREATE LOGIN POLICY policy-name policy-option
```

```
policy-option:
  policy-option-name = policy-option-value
```

```
policy-option-value:
  { UNLIMITED | DEFAULT | value }
```

```
policy-option-name:
  AUTO_UNLOCK_TIME
  | CHANGE_PASSWORD_DUAL_CONTROL
```

```
| DEFAULT_LOGICAL_SERVER  
| LOCKED  
| MAX_CONNECTIONS  
| MAX_DAYS_SINCE_LOGIN  
| MAX_FAILED_LOGIN_ATTEMPTS  
| MAX_NON_DBA_CONNECTIONS  
| PASSWORD_EXPIRY_ON_NEXT_LOGIN  
| PASSWORD_GRACE_TIME  
| PASSWORD_LIFE_TIME  
| ROOT_AUTO_UNLOCK_TIME  
| LDAP_PRIMARY_SERVER  
| LDAP_SECONDARY_SERVER  
| LDAP_AUTO_FAILBACK_PERIOD  
| LDAP_FAILOVER_TO_STD  
| LDAP_REFRESH_DN
```

Parameters

- **policy-name** – the name of the login policy. Specify root to modify the root login policy.
- **policy-option-name** – the name of the policy option. See *Login Policy Options* and *LDAP Login Policy Options* for details on each option.
- **policy-option-value** – the value assigned to the login policy option. If you specify UNLIMITED, no limits are used. If you specify DEFAULT, the default limits are used. See *Login Policy Options* and *LDAP Login Policy Options* for supported values for each option.

Applies to

Simplex and multiplex.

Examples

- **Example 1** – creates the Test1 login policy. This login policy has an unlimited password life and allows the user a maximum of five attempts to enter a correct password before the account is locked.

```
CREATE LOGIN POLICY Test1  
password_life_time=UNLIMITED  
max_failed_login_attempts=5;
```

Usage

If you do not specify a policy option, values for this login policy are taken from the root login policy. New policies do not inherit the MAX_NON_DBA_CONNECTIONS and ROOT_AUTO_UNLOCK_TIME policy options.

Permissions

Requires MANAGE ANY LOGIN POLICY system privilege.

The following system privileges can override the noted login policy options:

Exception System Privilege	Login Policy Option
SERVER OPERATOR or DROP CONNECTION system privilege	MAX_NON_DBA_CONNS MAX_CONNECTIONS
MANAGE ANY USER system privilege	LOCKED MAX_DAYS_SINCE_LOGIN

Login Policy Options

Available options for root and user-defined login policies.

Option	Description
AUTO_UNLOCK_TIME	<p>The time period after which locked accounts not granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined in any login policy, including the root login policy.</p> <ul style="list-style-type: none"> • Values – 0 – UNLIMITED • Default – UNLIMITED • Applies to – All users not granted the MANAGE ANY USER system privilege.
CHANGE_PASSWORD_DUAL_CONTROL	<p>Requires input from two users, each granted the CHANGE PASSWORD system privilege, to change the password of another user.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF • Applies to – All users.

Option	Description
DEFAULT_LOGICAL_SERVER	<p>If the connection string specifies no logical server, the user connects to the DEFAULT_LOGICAL_SERVER option specified in the user's login policy.</p> <ul style="list-style-type: none"> • Values – <ul style="list-style-type: none"> • Name of an existing user-defined logical server • ALL – allows access to all logical servers. • AUTO – value of the default logical server in the root login policy. • COORDINATOR – the current coordinator node. • NONE – denies access to any multiplex server. • OPEN – use alone or with the name of a user-defined logical server. Allows access to all multiplex nodes that are not members of any user-defined logical servers. • SERVER – allows access to all of the multiplex nodes, subject to the semantics of the SERVER logical server. • Default – AUTO • Applies to – All users. Requires MANAGE MULTIPLEX system privilege.
LOCKED	<p>If set ON, users cannot establish new connections. This setting temporarily denies access to login policy users. Logical server overrides for this option are not allowed.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF • Applies to – All users except those with the MANAGE ANY USER system privilege.
MAX_CONNECTIONS	<p>The maximum number of concurrent connections allowed for a user. You can specify a per-logical-server setting for this option.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users except those with the SERVER OPERATOR or DROP CONNECTION system privilege.

Option	Description
MAX_DAYS_SINCE_LOGIN	<p>The maximum number of days that can elapse between two successive logins by the same user.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users except those with the MANAGE ANY USER system privilege.
MAX_FAILED_LOGIN_ATTEMPTS	<p>The maximum number of failed attempts, since the last successful attempt, to log into the user account before the account is locked.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users.
MAX_NON_DBA_CONNECTIONS	<p>The maximum number of concurrent connections that a user without SERVER OPERATOR or DROP CONNECTION system privileges can make. This option is supported only in the root login policy.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users except those with the SERVER OPERATOR or DROP CONNECTION privilege.
PASSWORD_EXPIRY_ON_NEXT_LOGIN	<p>If set ON, the user's password expires at the next login.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – OFF • Applies to – All users. <hr/> <p>Note: This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL).</p>
PASSWORD_GRACE_TIME	<p>The number of days before password expiration during which login is allowed but the default post_login procedure issues warnings.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – 0 • Applies to – All users.

Option	Description
PASS-WORD_LIFE_TIME	<p>The maximum number of days before a password must be changed.</p> <ul style="list-style-type: none"> • Values – 0 – 2147483647 • Default – UNLIMITED • Applies to – All users.
ROOT_AUTO_UNLOCK_TIME	<p>The time period after which locked accounts granted the MANAGE ANY USER system privilege are automatically unlocked. This option can be defined only in the root login policy.</p> <ul style="list-style-type: none"> • Values – 0 – UNLIMITED • Default – 15 • Applies to – All users granted the MANAGE ANY USER system privilege.

LDAP Login Policy Options

Available login policy options for LDAP user authentication

Option	Description
LDAP_PRIMARY_SERVER	<p>Specifies the name of the primary LDAP server.</p> <ul style="list-style-type: none"> • Values – n/a • Default – None • Applies to – All users.
LDAP_SECONDARY_SERVER	<p>Specifies the name of the secondary LDAP server.</p> <ul style="list-style-type: none"> • Values – n/a • Default – None • Applies to – All users.
LDAP_AUTO_FAILBACK_PERIOD	<p>Specifies the time period, in minutes, after which automatic failback to the primary server is attempted.</p> <ul style="list-style-type: none"> • Values – 0 - 2147483647 • Default – 15 minutes • Applies to – All users.

Option	Description
LDAP_FAIL-OVER_TO_STD	<p>Permits authentication with standard authentication when authentication with the LDAP server fails due to system resources, network outage, connection timeouts, or similar system failures. However, it does not permit an actual authentication failure returned from an LDAP server to fail over to standard authentication.</p> <ul style="list-style-type: none"> • Values – ON, OFF • Default – ON • Applies to – All users.
LDAP_REFRESH_DN	<p>Updates the ldap_refresh_dn value in the ISYSLOGINPOLICYOPTION system table with the current time, stored in Coordinated Universal Time (UTC).</p> <p>Each time a user authenticates with LDAP, if the value of ldap_refresh_dn in ISYSLOGINPOLICYOPTION is more recent than the value of user_dn in ISYSUSER, a search for a new user DN occurs. The user_dn value is then updated with the new user DN and the user_dn_changed_at value is again updated to the current time.</p> <ul style="list-style-type: none"> • Values – NOW • Initial value for ROOT policy – NULL • Initial value for user-defined login policy – Current time stored in UTC • Applies to – All users.

Multiplex Login Policy Configuration

Configure login policies for multiplex servers.

Example

This example overrides the login policy settings on a logical server, increasing the maximum number of connections on logical server `ls1`:

```
ALTER LOGIN POLICY lp1 max_connections=20 LOGICAL SERVER ls1;
```

Usage

Applies only to multiplex.

Any login management commands you execute on any multiplex server automatically propagate to all servers in the multiplex. For best performance, execute these commands, or any DDL, on the coordinator.

An override at the logical server level override means that a particular login policy option has different settings for different logical servers. `SYS.ISYSIQLSLOGINPOLICYOPTION` stores login policy option values for logical-server override. For each logical-server override

of a login policy option, a corresponding row exists in `ISYSIQLSLOGINPOLICYOPTION`.

CREATE LS POLICY Statement

Creates a user-defined logical server policy. This statement enforces consistent shared system temporary store settings across physical nodes shared by logical servers.

Syntax

```
CREATE LS POLICY ls-policy-name ls-option-value-list [ WITH STOP  
SERVER ]
```

```
ls-option-value-list:  
  { ls-option-name = ls-policy-option-value } ...
```

```
ls-option-name:  
  ALLOW_COORDINATOR_AS_MEMBER  
  | DQP_ENABLED  
  | LOGIN_REDIRECTION  
  | REDIRECTION_WAITERS_THRESHOLD  
  | TEMP_DATA_IN_SHARED_TEMP
```

Parameters

- **ls-policy-name** – the name of the logical server policy. You can specify any identifier except root for the policy name.
- **ls-option-value-list** – the name of the logical server policy option. See *LS Policy Options* for details on each option.
- **ls-policy-option-value** – any unspecified option inherits its value from the root logical server policy. See *LS Policy Options* for supported values for each option.
- **WITH STOP SERVER** – automatically shuts down all servers in the logical server when the `TEMP_DATA_IN_SHARED_TEMP` option is changed directly or indirectly.

Applies to

Multiplex only.

Examples

- **Example 1** – creates a user-defined logical server policy named *lspolicy1*:

```
CREATE LS POLICY lspolicy1  
ALLOW_COORDINATOR_AS_MEMBER=ON;
```

Usage

If you want a smaller `IQ_SYSTEM_TEMP` dbspace, set `TEMP_DATA_IN_SHARED_TEMP` to `ON`, which writes temporary data to `IQ_SHARED_TEMP` instead of `IQ_SYSTEM_TEMP`. In a distributed query processing environment, however, setting both `DQP_ENABLED` and `TEMP_DATA_IN_SHARED_TEMP` to `ON` may saturate your SAN with additional data in `IQ_SHARED_TEMP`, where additional I/O operations against `IQ_SHARED_TEMP` may adversely affect DQP performance.

Standards

- SQL – vendor extension to ISO/ANSI SQL grammar.
- Sybase – not supported by Adaptive Server Enterprise.

Permissions

Requires the `MANAGE MULTIPLEX` system privilege.

CREATE MESSAGE Statement [T-SQL]

Adds a user-defined message to the `SYSUSERMESSAGES` system table for use by **PRINT** and **RAISERROR** statements.

Syntax

```
CREATE MESSAGE message-number
... AS 'message-text'
```

Parameters

- **message_number** – the message number of the message to add. The message number for a user-defined message must be 20000 or greater.
- **message_text** – the text of the message to add. The maximum length is 255 bytes. **PRINT** and **RAISERROR** recognize placeholders in the message text to print out. A single message can contain up to 20 unique placeholders in any order. These placeholders are replaced with the formatted contents of any arguments that follow the message when the text of the message is sent to the client.

Placeholders are numbered to allow reordering of the arguments when translating a message to a language with a different grammatical structure. A placeholder for an argument appears as “%nn!”—a percent sign (%), followed by an integer from 1 to 20, followed by an exclamation mark (!)—where the integer represents the position of the argument in the argument list, “%1!” is the first argument, “%2!” is the second argument, and so on.

Usage

CREATE MESSAGE associates a message number with a message string. The message number can be used in **PRINT** and **RAISERROR** statements.

There is no parameter corresponding to the *language* argument for **sp_addmessage**.

Side Effects

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—The functionality of **CREATE MESSAGE** is provided by the **sp_addmessage** procedure in Adaptive Server Enterprise.

Permissions

Requires one of:

- CREATE MESSAGE system privilege.
- CREATE ANY OBJECT system privilege.

See also

- *PRINT Statement [T-SQL]* on page 378
- *RAISERROR Statement [T-SQL]* on page 381

CREATE MULTIPLEX SERVER Statement

Creates a multiplex server.

Syntax

```
CREATE MULTIPLEX SERVER server-name DATABASE  
  'dbfile' host-port list [ ROLE { READER | WRITER } ]  
  [ STATUS | { INCLUDED | EXCLUDED } ]
```

host-port-list:

```
{ [ PRIVATE ] HOST 'hostname' PORT port number }
```

Parameters

- **PRIVATE** – specifies that the particular HOST PORT pair is for private interconnection. A separate private interconnection for multiplex interprocess communication (MIPC) enables highly available and high-performance network configurations. SAP Sybase IQ

automatically opens private ports; you need not list them in the host-port-list used to start the server. All public and private ports require unique port numbers to avoid conflicts.

- **server-name** – the name of the multiplex server based on the rules for server startup option -n.
- **ROLE** – default if not specified is **READER**.
- **STATUS** – default if not specified is **INCLUDED**.

Applies to

Multiplex only.

Usage

If you plan to use UNIX soft (symbolic) links for server paths, create the soft link before you run **CREATE MULTIPLEX SERVER**. When you start the new server, the database file path must match the database file path specified when creating that server.

When creating the initial multiplex server, both coordinator node and secondary node rows are added to `SYS.ISYSIQMPXSERVER`. The transaction log records this operation as two separate **CREATE MULTIPLEX SERVER** commands, one for the coordinator node and one for the secondary node.

After creating the first secondary node, the coordinator shuts down automatically.

The `SYS.ISYSIQMPXSERVER` system table stores the `HOST` hostname `PORT` portname pairs in its `connection_info` string as `host:port[:host:port...]`.

Note: Use multiple `host:port` pairs if the computer the multiplex server is running on has multiple redundant network cards mapped to different network addresses.

You may specify the clauses `DATABASE`, host-port list, `ROLE` and `STATUS` in any order.

When you add a server, the coordinator must be running, but you can run the **CREATE MULTIPLEX SERVER** command from any server in the multiplex.

This statement is automatically committed.

Permissions

Requires the `MANAGE MULTIPLEX` system privilege.

CREATE PROCEDURE Statement

Creates a new user-defined SQL procedure in the database.

To create external procedure interfaces, see *CREATE PROCEDURE Statement (External Procedures)*.

Syntax

```

CREATE [ OR REPLACE | TEMPORARY ] PROCEDURE [ owner.]procedure-name
( [ parameter, ... ] ) {
  [ SQL SECURITY { INVOKER | DEFINER } ]
  [ RESULT ( result-column, ... ) | NO RESULT SET ]
  [ ON EXCEPTION RESUME ] compound statement | AT location-string

parameter:
  parameter_mode parameter-name data-type [ DEFAULT expression ] |
SQLCODE | SQLSTATE

parameter mode:
  IN | OUT | INOUT

result-column:
  column-name data-type

```

Parameters

- **parameter-name** – parameter names must conform to the rules for other database identifiers, such as column names, and must be a valid SQL data type. The keywords have the following meanings:

Parameters can be prefixed by one of the keywords IN, OUT or INOUT. If no keyword is specified, parameters are INOUT by default. The keywords have the following meanings:

- **IN** – parameter is an expression that provides a value to the procedure.
- **OUT** – parameter is a variable that could be given a value by the procedure.
- **INOUT** – parameter is a variable that provides a value to the procedure, and could be given a new value by the procedure.
- **SQLSTATE and SQLCODE** – special parameters that output the SQLSTATE or SQLCODE value when the procedure ends (they are OUT parameters). Whether or not a SQLSTATE and SQLCODE parameter is specified, the SQLSTATE and SQLCODE special values can always be checked immediately after a procedure call to test the return status of the procedure.

The SQLSTATE and SQLCODE special values are modified by the next SQL statement. Providing SQLSTATE or SQLCODE as procedure arguments allows the return code to be stored in a variable.

- **CREATE** – creates a new procedure.
- **OR REPLACE** – replaces an existing procedure with the same name. This clause changes the definition of the procedure, but preserves existing permissions.

You cannot use the OR REPLACE clause with temporary procedures. Also, an error is returned if the procedure being replaced is already in use.

- **TEMPORARY** – the stored procedure is visible only by the connection that created it, and that it is automatically dropped when the connection is dropped. You can also explicitly

drop temporary stored procedures. You cannot perform **ALTER**, **GRANT**, or **REVOKE** on them, and, unlike other stored procedures, temporary stored procedures are not recorded in the catalog or transaction log.

Temporary procedures execute with the permissions of their creator (current user), or specified owner. You can specify an owner for a temporary procedure when:

- The temporary procedure is created within a permanent stored procedure
- The temporary and permanent procedure both have the same owner

To drop the owner of a temporary procedure, drop the temporary procedure first.

You can create and drop temporary stored procedures when you are connected to a read-only database; they cannot be external procedures.

For example, the following temporary procedure drops the table called `CustRank`, if it exists. For this example, the procedure assumes that the table name is unique and can be referenced by the procedure creator without specifying the table owner:

```
CREATE TEMPORARY PROCEDURE drop_table( IN @TableName char(128) )
BEGIN
    IF EXISTS ( SELECT 1 FROM SYS.SYSTAB WHERE
        table_name = @TableName )
    THEN EXECUTE IMMEDIATE
        'DROP TABLE "' || @TableName || "'';
    MESSAGE 'Table "' || @TableName ||
        '" dropped' to client;
    END IF;
END;
CALL drop_table( 'CustRank' )
```

- **RESULT** – declares the number and type of columns in the result set. The parenthesized list following the **RESULT** keyword defines the result column names and types. This information is returned by the Embedded SQL `DESCRIBE` or by ODBC `SQLDescribeCol` when a **CALL** statement is being described. Allowed data types are listed in *Reference: Building Blocks, Tables, and Procedures > SQL Data Types*.

Some procedures can produce more than one result set, depending on how they are executed. For example, this procedure returns two columns under some circumstances, and one in others.

```
CREATE PROCEDURE names( IN formal char(1))
BEGIN
    IF formal = 'n' THEN
        SELECT GivenName
        FROM Employees
    ELSE
        SELECT Surname, GivenName
        FROM Employees
    END IF
END
```

Procedures with variable result sets must be written without a **RESULT** clause, or in Transact-SQL. Their use is subject to these limitations:

- **Embedded SQL** – you must DESCRIBE the procedure call after the cursor for the result set is opened, but before any rows are returned, in order to get the proper shape of result set. The CURSOR *cursor-name* clause on the DESCRIBE statement is required.
- **ODBC, OLE DB, ADO.NET** – variable result-set procedures can be used by ODBC applications. The proper description of the result sets is carried out by the driver or provider.
- **Open Client applications** – variable result-set procedures can be used by Open Client applications.

If your procedure returns only one result set, use a RESULT clause. The presence of this clause prevents ODBC and Open Client applications from describing the result set again after a cursor is open.

To handle multiple result sets, ODBC must describe the currently executing cursor, not the procedure's defined result set. Therefore, ODBC does not always describe column names as defined in the RESULT clause of the procedure definition. To avoid this problem, use column aliases in the SELECT statement that generates the result set.

- **NO RESULT SET** – declares that this procedure returns no result set. This is useful when an external environment needs to know that a procedure does not return a result set.
- **SQL SECURITY** – defines whether the procedure is executed as the INVOKER (the user who is calling the procedure), or as the DEFINER (the user who owns the procedure). The default is DEFINER.

Extra memory is used when you specify SQL SECURITY INVOKER, because annotation must be done for each user that calls the procedure. Also, name resolution is performed as the invoker as well. Therefore, qualify all object names (tables, procedures, and so on) with their appropriate owner. For example, suppose `user1` creates this procedure:

```
CREATE PROCEDURE user1.myProcedure ()
  RESULT( columnA INT )
  SQL SECURITY INVOKER
  BEGIN
    SELECT columnA FROM table1;
  END;
```

If `user2` attempts to run this procedure and a table `user2.table1` does not exist, a table lookup error results. Additionally, if a `user2.table1` does exist, that table is used instead of the intended `user1.table1`. To prevent this situation, qualify the table reference in the statement (`user1.table1`, instead of just `table1`).

- **ON EXCEPTION RESUME** – the procedure takes an action that depends on the setting of the **ON_TSQL_ERROR** option. If **ON_TSQL_ERROR** option is set to **CONDITIONAL** (which is the default) the execution continues if the next statement handles the error; otherwise, it exits.

Error-handling statements include:

- **IF**
- **SELECT** @variable =
- **CASE**
- **LOOP**
- **LEAVE**
- **CONTINUE**
- **CALL**
- **EXECUTE**
- **SIGNAL**
- **RESIGNAL**
- **DECLARE**
- **SET VARIABLE**

Do not use explicit error-handling code with an ON EXCEPTION RESUME clause.

See *ON_TSQL_ERROR Option [TSQL]*.

- **AT location-string** – creates a *proxy stored procedure* on the current database for a remote procedure specified by *location-string*. The AT clause supports the semicolon (;) as a field delimiter in *location-string*. If no semicolon is present, a period is the field delimiter. This allows file names and extensions to be used in the database and owner fields.

Examples

- **Example 1** – uses a case statement to classify the results of a query:

```
CREATE PROCEDURE ProductType (IN product_id INT, OUT type
CHAR(10))
BEGIN
  DECLARE prod_name CHAR(20) ;
  SELECT name INTO prod_name FROM "GROUPO"."Products"
  WHERE ID = product_id;
  CASE prod_name
  WHEN 'Tee Shirt' THEN
    SET type = 'Shirt'
  WHEN 'Sweatshirt' THEN
    SET type = 'Shirt'
  WHEN 'Baseball Cap' THEN
    SET type = 'Hat'
  WHEN 'Visor' THEN
    SET type = 'Hat'
  WHEN 'Shorts' THEN
    SET type = 'Shorts'
  ELSE
    SET type = 'UNKNOWN'
  END CASE ;
END
```

- **Example 2** – uses a cursor and loop over the rows of the cursor to return a single value:

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
```

SQL Statements

```
BEGIN
  DECLARE err_notfound EXCEPTION
    FOR SQLSTATE '02000' ;
  DECLARE curThisCust CURSOR FOR
  SELECT CompanyName, CAST(          sum(SalesOrderItems.Quantity *
  Products.UnitPrice) AS INTEGER) VALUE
  FROM Customers
  LEFT OUTER JOIN SalesOrders
  LEFT OUTER JOIN SalesorderItems
  LEFT OUTER JOIN Products
  GROUP BY CompanyName ;

  DECLARE ThisValue INT ;
  DECLARE ThisCompany CHAR(35) ;
  SET TopValue = 0 ;
  OPEN curThisCust ;
  CustomerLoop:
  LOOP
    FETCH NEXT curThisCust
    INTO ThisCompany, ThisValue ;
    IF SQLSTATE = err_notfound THEN
      LEAVE CustomerLoop ;
    END IF ;
    IF ThisValue > TopValue THEN
      SET TopValue = ThisValue ;
      SET TopCompany = ThisCompany ;
    END IF ;
  END LOOP CustomerLoop ;
  CLOSE curThisCust ;
END
```

Usage

CREATE PROCEDURE creates a procedure in the database. A procedure is invoked with a **CALL** statement. You can create permanent or temporary (TEMPORARY) stored procedures. You can use PROC as a synonym for PROCEDURE.

Note: There are two ways to create stored procedures: ISO/ANSI SQL and T-SQL. **BEGIN TRANSACTION**, for example, is T-SQL-specific when using **CREATE PROCEDURE** syntax. Do not mix syntax when creating stored procedures. See *CREATE PROCEDURE Statement [T-SQL]*.

When procedures are executed using **CALL**, not all parameters need to be specified. If a default value is provided in the **CREATE PROCEDURE** statement, missing parameters are assigned the default values. If an argument is not provided in the **CALL** statement, and no default is set, an error is given.

Remote procedures can return only up to 254 characters in output variables.

If a remote procedure can return a result set, even if it does not return one in all cases, then the local procedure definition must contain a RESULT clause.

For information on remote servers, see *CREATE SERVER Statement*.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—The Transact-SQL **CREATE PROCEDURE** statement is different.
- SQLJ—The syntax extensions for Java result sets are as specified in the proposed SQLJ1 standard.

Permissions

Watcom SQL or Transact SQL procedure to be owned by self – Requires CREATE PROCEDURE system privilege.

Watcom SQL or Transact SQL procedure to be owned by any user – Requires one of:

- CREATE ANY PROCEDURE system privilege.
- CREATE ANY OBJECT system privilege.

Remote procedure to be owned by self – Requires all of:

- CREATE EXTERNAL REFERENCE system privilege.
- CREATE PROCEDURE system privilege.

Remote procedure to be owned by any user – Requires CREATE EXTERNAL REFERENCE system privilege. Also requires one of:

- CREATE ANY PROCEDURE system privilege.
- CREATE ANY OBJECT system privilege.

See also

- *Referencing Temporary Tables Within Procedures* on page 172
- *BEGIN ... END Statement* on page 84
- *CALL Statement* on page 90
- *CREATE PROCEDURE Statement [T-SQL]* on page 172
- *CREATE PROCEDURE Statement (External Procedures)* on page 175
- *CREATE SERVER Statement* on page 192
- *DROP Statement* on page 255
- *EXECUTE IMMEDIATE Statement [ESQL] [SP]* on page 279
- *GRANT EXECUTE Statement* on page 310
- *RAISERROR Statement [T-SQL]* on page 381
- *ON_TSQL_ERROR Option [TSQL]* on page 598

Referencing Temporary Tables Within Procedures

Sharing a temporary table between procedures can cause problems if the table definitions are inconsistent.

For example, suppose you have two procedures `procA` and `procB`, both of which define a temporary table, `temp_table`, and call another procedure called `sharedProc`. Neither `procA` nor `procB` has been called yet, so the temporary table does not yet exist.

Now, suppose that the `procA` definition for `temp_table` is slightly different than the definition in `procB`—while both used the same column names and types, the column order is different.

When you call `procA`, it returns the expected result. However, when you call `procB`, it returns a different result.

This is because when `procA` was called, it created `temp_table`, and then called `sharedProc`. When `sharedProc` was called, the **SELECT** statement inside of it was parsed and validated, and then a parsed representation of the statement is cached so that it can be used again when another **SELECT** statement is executed. The cached version reflects the column ordering from the table definition in `procA`.

Calling `procB` causes the `temp_table` to be recreated, but with different column ordering. When `procB` calls `sharedProc`, the database server uses the cached representation of the **SELECT** statement. So, the results are different.

You can avoid this from happening by doing one of the following:

- ensure that temporary tables used in this way are defined consistently
- consider using a global temporary table instead

CREATE PROCEDURE Statement [T-SQL]

Creates a new procedure that is compatible with Adaptive Server Enterprise.

Syntax

This subset of the Transact-SQL **CREATE PROCEDURE** statement is supported in SAP Sybase IQ:

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure_name
... [ [ ( ) @parameter_name data-type [ = default ] [ OUTPUT ] [ , ... ]
[ ) ] ]
...[ WITH RECOMPILE ]
... AS
... statement-list
```

Parameters

- **CREATE** – creates a new procedure.
- **OR REPLACE** – replaces an existing procedure with the same name. This clause changes the definition of the procedure, but preserves existing permissions.

Usage

Differences between Transact-SQL and SAP Sybase IQ SQL statements:

- **Variable names prefixed by @** – the “@” sign denotes a Transact-SQL variable name; SAP Sybase IQ variables can be any valid identifier and the @ prefix is optional.
- **Input and output parameters** – SAP Sybase IQ procedure parameters are specified as IN, OUT, or INOUT; Transact-SQL procedure parameters are INPUT parameters by default or can be specified as OUTPUT. Those parameters declared as INOUT or as OUT in SAP Sybase IQ should be declared with OUTPUT in Transact-SQL.
- **Parameter default values** – SAP Sybase IQ procedure parameters are given a default value using the keyword DEFAULT; Transact-SQL uses an equality sign (=) to provide the default value.
- **Returning result sets** – SAP Sybase IQ uses a RESULT clause to specify returned result sets. In Transact-SQL procedures, the column names or alias names of the first query are returned to the calling environment:

```
CREATE PROCEDURE showdept @deptname varchar(30)
AS
    SELECT Employees.Surname, Employees.givenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = @deptname
    AND Departments.DepartmentID =
        Employees.DepartmentID
```

The corresponding SAP Sybase IQ procedure:

```
CREATE PROCEDURE showdept(in deptname
    varchar(30) )
RESULT ( lastname char(20), firstname char(20))
ON EXCEPTION RESUME
BEGIN
    SELECT Employees.SurName, Employees.GivenName
    FROM Departments, Employees
    WHERE Departments.DepartmentName = deptname
    AND Departments.DepartmentID =
        Employees.DepartmentID
END
```

- **Procedure body** – the body of a Transact-SQL procedure is a list of Transact-SQL statements prefixed by the AS keyword. The body of an SAP Sybase IQ procedure is a compound statement, bracketed by BEGIN and END keywords.

Note: There are two ways to create stored procedures: T-SQL and SQL/92. **BEGIN TRANSACTION**, for example, is T-SQL specific when using **CREATE PROCEDURE** syntax. Do not mix syntax when creating stored procedures.

Side Effects

- Automatic commit

Standards

- SQL—Transact-SQL extension to ISO/ANSI SQL grammar.
- Sybase—SAP Sybase IQ supports a subset of the Adaptive Server Enterprise **CREATE PROCEDURE** statement syntax.

If the Transact-SQL **WITH RECOMPILE** optional clause is supplied, it is ignored. SQL Anywhere always recompiles procedures the first time they are executed after a database is started, and stores the compiled procedure until the database is stopped.

Groups of procedures are not supported.

Permissions

Watcom SQL or Transact SQL procedure to be owned by self – Requires **CREATE PROCEDURE** system privilege.

Watcom SQL or Transact SQL procedure to be owned by any user – Requires one of:

- **CREATE ANY PROCEDURE** system privilege.
- **CREATE ANY OBJECT** system privilege.

Remote procedure to be owned by self – Requires all of:

- **CREATE EXTERNAL REFERENCE** system privilege.
- **CREATE PROCEDURE** system privilege.

Remote procedure to be owned by any user – Requires **CREATE EXTERNAL REFERENCE** system privilege. Also requires one of:

- **CREATE ANY PROCEDURE** system privilege.
- **CREATE ANY OBJECT** system privilege.

See also

- *CREATE PROCEDURE Statement* on page 165

CREATE PROCEDURE Statement (External Procedures)

Creates an interface to a native or external procedure.

For **CREATE PROCEDURE** reference information for Java UDFs, see *CREATE PROCEDURE Statement (Java UDF)*. For **CREATE PROCEDURE** reference information for table UDFs, see *CREATE PROCEDURE Statement (Table UDF)*

Syntax

```
CREATE [ OR REPLACE ] PROCEDURE [ owner.]procedure-name ( [ parameter,
... ] )
  [ SQL SECURITY { INVOKER | DEFINER } ]
  [ RESULT ( result-column, ...) | NO RESULT SET ]
  [ DYNAMIC RESULT SETS integer-expression ]
  [ EXTERNAL NAME 'native-call'
    | EXTERNAL NAME 'c-call' LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32
    | C_ODBC64 }
    | EXTERNAL NAME 'perl-call' LANGUAGE PERL
    | EXTERNAL NAME 'php-call' LANGUAGE PHP
    | EXTERNAL NAME 'java-call' LANGUAGE JAVA ]
```

parameter:
parameter_mode parameter-name data-type [DEFAULT expression] |
SQLCODE | **SQLSTATE**

parameter_mode:
IN | **OUT** | **INOUT**

result-column:
column-name data-type

native-call:
[operating-system:]function-name@library

c-call:
[operating-system:]function-name@library; ...

perl-call:
<file=perl-file> \$sa_perl_return = perl-subroutine(\$sa_perl_arg0[, ...])

php-call:
<file=php-file> print php-func(\$argv[1][, ...])

java-call:
[package-name.]class-name.method-name method-signature

operating-system:
Unix

method-signature:
([field-descriptor, ...]) return-descriptor

```

field-descriptor and return-descriptor:
{ Z
  | B
  | S
  | I
  | J
  | F
  | D
  | C
  | V
  | [descriptor
  | Lclass-name;
}

```

Parameters

- **CREATE** – creates a new procedure.
- **OR REPLACE** – replaces an existing procedure with the same name. This clause changes the definition of the procedure, but preserves existing permissions.
- **parameter** – parameter names must conform to the rules for other database identifiers, such as column names, and must be a valid SQL data type. The keywords have the following meanings:

Parameters can be prefixed by one of the keywords **IN**, **OUT** or **INOUT**. If no keyword is specified, parameters are **INOUT** by default. The keywords have the following meanings:

- **IN** – parameter is an expression that provides a value to the procedure.
- **OUT** – parameter is a variable that could be given a value by the procedure.
- **INOUT** – parameter is a variable that provides a value to the procedure, and could be given a new value by the procedure.

Note: TABLE parameters cannot be declared as **INOUT** or **OUT**. See *CREATE PROCEDURE Statement (Table UDF)*.

When procedures are executed using **CALL**, not all parameters need to be specified. If a default value is provided in the **CREATE PROCEDURE** statement, missing parameters are assigned the default values. If an argument is not provided in the **CALL** statement, and no default is set, an error is given.

Note: You cannot **CALL** a table UDF. Use the **CREATE PROCEDURE** statement.

- **RESULT** – declares the number and type of columns in the result set. The parenthesized list following the **RESULT** keyword defines the result column names and types. This information is returned by the Embedded SQL **DESCRIBE** or by ODBC **SQLDescribeCol** when a **CALL** statement is being described. Allowed data types are listed in *Reference: Building Blocks, Tables, and Procedures > SQL Data Types*.

Embedded SQL (LANGUAGE C_ESQL32, LANGUAGE C_ESQL64) or ODBC (LANGUAGE C_ODBC32, LANGUAGE C_ODBC64) external procedures can return 0 or 1 result sets.

Perl or PHP (LANGUAGE PERL, LANGUAGE PHP) external procedures cannot return result sets. Procedures that call native functions loaded by the database server cannot return result sets.

CLR or Java (LANGUAGE CLR, LANGUAGE JAVA) external procedures can return 0, 1, or more result sets.

- **NO RESULT SET** – declares that this procedure returns no result set. This is useful when an external environment needs to know that a procedure does not return a result set.
- **DYNAMIC RESULT SETS** – Use this clause with LANGUAGE CLR and LANGUAGE JAVA calls. This clause is meaningful only if you specify LANGUAGE. If you specify a RESULT clause, DYNAMIC RESULT SETS defaults to 1. If you do not specify a RESULT clause, DYNAMIC RESULT SETS defaults to 0. Note that procedures that call into Perl or PHP (LANGUAGE PERL, LANGUAGE PHP) external functions cannot return result sets. Procedures that call native functions loaded by the database server cannot return result sets.

The C_ESQL32, C_ESQL64, C_ODBC32, and C_ODBC64 external environments can also return result sets (like CLR and JAVA), but they are restricted to only one dynamic result set.

Procedures that call into Perl or PHP (LANGUAGE PERL, LANGUAGE PHP) external functions cannot return result sets. Procedures that call native functions loaded by the database server cannot return result sets.

- **SQL SECURITY** – defines whether the procedure is executed as the INVOKER (the user who is calling the procedure), or as the DEFINER (the user who owns the procedure). The default is DEFINER. For external calls, this clause establishes the ownership context for unqualified object references in the external environment.

Extra memory is used when you specify SQL SECURITY INVOKER, because annotation must be done for each user that calls the procedure. Also, name resolution is performed as the invoker as well. Therefore, qualify all object names (tables, procedures, and so on) with their appropriate owner. For example, suppose `user1` creates this procedure:

```
CREATE PROCEDURE user1.myProcedure ()
  RESULT( columnA INT )
  SQL SECURITY INVOKER
  BEGIN
    SELECT columnA FROM table1;
  END;
```

If `user2` attempts to run this procedure and a table `user2.table1` does not exist, a table lookup error results. Additionally, if a `user2.table1` does exist, that table is used

instead of the intended `user1.table1`. To prevent this situation, qualify the table reference in the statement (`user1.table1`, instead of just `table1`).

- **EXTERNAL NAME** – a procedure using the **EXTERNAL NAME** clause with no **LANGUAGE** attribute defines an interface to a native function written in a programming language such as C. The native function is loaded by the database server into its address space.

The library name can include the file extension, which is typically `.dll` on Windows and `.so` on UNIX. In the absence of the extension, the software appends the platform-specific default file extension for libraries. This is a formal example:

```
CREATE PROCEDURE mystring( IN instr LONG VARCHAR )
EXTERNAL NAME
'mystring@mylib.dll;Unix:mystring@mylib.so';
```

A simpler way to write the preceding **EXTERNAL NAME** clause, using platform-specific defaults:

```
CREATE PROCEDURE mystring( IN instr LONG VARCHAR )
EXTERNAL NAME 'mystring@mylib';
```

When called, the library containing the function is loaded into the address space of the database server. The native function executes as part of the server. In this case, if the function causes a fault, then the database server terminates. Because of this, loading and executing functions in an external environment using the **LANGUAGE** attribute is recommended. If a function causes a fault in an external environment, the database server continues to run.

- **EXTERNAL NAME c-call LANGUAGE { C_ESQL32 | C_ESQL64 | C_ODBC32 | C_ODBC64 }** : – to call a compiled native C function in an external environment instead of within the database server, the stored procedure or function is defined with the **EXTERNAL NAME** clause followed by the **LANGUAGE** attribute specifying one of `C_ESQL32`, `C_ESQL64`, `C_ODBC32`, or `C_ODBC64`.

When the **LANGUAGE** attribute is specified, then the library containing the function is loaded by an external process and the external function will execute as part of that external process. In this case, if the function causes a fault, then the database server will continue to run.

The following is a sample procedure definition.

```
CREATE PROCEDURE ODBCinsert(
    IN ProductName CHAR(30),
    IN ProductDescription CHAR(50)
)
NO RESULT SET
EXTERNAL NAME 'ODBCexternalInsert@extodbc.dll'
LANGUAGE C_ODBC32;
```

- **EXTERNAL NAME perl-call LANGUAGE CLR** – to call a Perl function in an external environment, the procedure interface is defined with an **EXTERNAL NAME** clause followed by the **LANGUAGE PERL** attribute.

A Perl stored procedure or function behaves the same as a SQL stored procedure or function with the exception that the code for the procedure or function is written in Perl and the execution of the procedure or function takes place outside the database server (that is, within a Perl executable instance).

Sample procedure definition:

```
CREATE PROCEDURE PerlWriteToConsole( IN str LONG VARCHAR)
NO RESULT SET
EXTERNAL NAME '<file=PerlConsoleExample>
WriteToServerConsole( $sa_perl_arg0 )'
LANGUAGE PERL;
```

- **EXTERNAL NAME perl-call LANGUAGE PHP** – to call a PHP function in an external environment, the procedure interface is defined with an **EXTERNAL NAME** clause followed by the **LANGUAGE PHP** attribute.

A PHP stored procedure or function behaves the same as a SQL stored procedure or function with the exception that the code for the procedure or function is written in PHP and the execution of the procedure or function takes place outside the database server (that is, within a PHP executable instance).

Sample procedure definition:

```
CREATE PROCEDURE PHPPopulateTable()
NO RESULT SET
EXTERNAL NAME '<file=ServerSidePHPExample>
ServerSidePHPSub()'
LANGUAGE PHP;
```

- **EXTERNAL NAME java-call LANGUAGE JAVA** – A Java method signature is a compact character representation of the types of the parameters and the type of the return value.

To call a Java method in an external environment, the procedure interface is defined with an **EXTERNAL NAME** clause followed by the **LANGUAGE JAVA** attribute.

A Java-interfacing stored procedure or function behaves the same as a SQL stored procedure or function with the exception that the code for the procedure or function is written in Java and the execution of the procedure or function takes place outside the database server (that is, within a Java Virtual Machine).

Sample procedure definition:

```
CREATE PROCEDURE HelloDemo( IN
name LONG VARCHAR )
NO RESULT SET
EXTERNAL NAME 'Hello.main([Ljava/lang/String;)V'
LANGUAGE JAVA;
```

The descriptors for arguments and return values from Java methods have the following meanings:

Field Type	Java Data Type
B	byte
C	char
D	double
F	float
I	int
J	long
L <i>class-name</i> ;	an instance of the <i>class-name</i> class. The class name must be fully qualified, and any dot in the name must be replaced by a backslash. For example, java/lang/String
S	short
V	void
Z	boolean
[use one for each dimension of an array

For example:

```
double some_method(
    boolean a,
    int b,
    java.math.BigDecimal c,
    byte [][] d,
    java.sql.ResultSet[] d ) {
}
```

has the signature:

```
'(ZILjava/math/BigDecimal; [[B[Ljava/sql/ResultSet;)D'
```

Usage

The body of a procedure consists of a compound statement. For information on compound statements, see *BEGIN ... END Statement*.

Note: There are two ways to create stored procedures: ISO/ANSI SQL and T-SQL. **BEGIN TRANSACTION**, for example, is T-SQL specific when using **CREATE PROCEDURE** syntax. Do not mix syntax when creating stored procedures. See *CREATE PROCEDURE Statement [T-SQL]*.

If a stored procedure returns a result set, it cannot also set output parameters or return a return value.

You cannot create **TEMPORARY** external call procedures.

When referencing a temporary table from multiple procedures, a potential issue can arise if the temporary table definitions are inconsistent and statements referencing the table are cached.

You can create permanent stored procedures that call external or native procedures written in a variety of programming languages. You can use **PROC** as a synonym for **PROCEDURE**.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—The Transact-SQL **CREATE PROCEDURE** statement is different.
- SQLJ—The syntax extensions for Java result sets are as specified in the proposed SQLJ1 standard.

Permissions

External procedure to be owned by self – Requires:

- CREATE EXTERNAL REFERENCE system privilege.
- CREATE PROCEDURE system privilege.

External procedure to be owned by any user – Requires CREATE EXTERNAL REFERENCE system privilege. Also requires one of:

- CREATE ANY PROCEDURE system privilege.
- CREATE ANY OBJECT system privilege.

See also

- *ALTER PROCEDURE Statement* on page 36
- *BEGIN ... END Statement* on page 84
- *CALL Statement* on page 90
- *CREATE PROCEDURE Statement* on page 165
- *CREATE PROCEDURE Statement [T-SQL]* on page 172
- *DROP Statement* on page 255
- *EXECUTE IMMEDIATE Statement [ESQL] [SP]* on page 279
- *GRANT EXECUTE Statement* on page 310

CREATE PROCEDURE Statement (Java UDF)

Creates an interface to an external Java table UDF.

For **CREATE PROCEDURE** reference information for external procedures, see *CREATE PROCEDURE Statement (External Procedures)*. For **CREATE PROCEDURE** reference information for table UDFs, see *CREATE PROCEDURE Statement (Table UDF)*

Syntax

Syntax 1 – For a query referencing at least one SAP Sybase IQ table:

```

CREATE[ OR REPLACE ] PROCEDURE
  [ owner.]procedure-name ( [ parameter, ...] )
  [ RESULT (result-column, ...)]
  [ SQL SECURITY { INVOKER | DEFINER } ]
  EXTERNAL NAME 'java-call' [ LANGUAGE java ] }

parameter:
  [ IN parameter_mode parameter-name data-type
  [ DEFAULT expression ]

result-column:
  column-name data-type

java:
  [ ALLOW | DISALLOW SERVER SIDE REQUESTS ]

java-call:
  '[ package-name.]class-name.method-name method-signature'

```

Syntax 2 – For a query referencing catalog store tables only:

```

CREATE[ OR REPLACE ] PROCEDURE
  [ owner.]procedure-name ( [ parameter, ...] )
  [ RESULT (result-column, ...)]
  | NO RESULT SET
  [ DYNAMIC RESULT SETS integer-expression ]
  [ SQL SECURITY { INVOKER | DEFINER } ]
  EXTERNAL NAME 'java-call' [ LANGUAGE Java ] }

parameter:
  [ IN | OUT | INOUT ] parameter_mode parameter-name data-type
  [ DEFAULT expression ]

result-column:
  column-name data-type

java:
  [ ALLOW | DISALLOW SERVER SIDE REQUESTS ]

java-call:
  '[ package-name.]class-name.method-name method-signature'

```

Parameters

- **java** – **DISALLOW** is the default. **ALLOW** indicates that server-side connections are allowed.

Note: Do not specify **ALLOW** unless necessary. A setting of **ALLOW** slows down certain types of SAP Sybase IQ table joins. If you change a procedure definition from **ALLOW** to **DISALLOW**, or vice-versa, the change will not be recognized until you make a new connection.

Do not use UDFs with both ALLOW SERVER SIDE REQUESTS and DISALLOW SERVER SIDE REQUESTS in the same query.

Usage

If your query references SAP Sybase IQ tables, note that different syntax and parameters apply compared to a query that references only catalog store tables.

Java table UDFs are only supported in the **FROM** clause.

For Java table functions, exactly one result set is allowed. If the Java table functions are joined with an SAP Sybase IQ table or if a column from an SAP Sybase IQ table is an argument to the Java table function then only one result set is supported.

If the Java table function is the only item in the FROM clause then N number of result sets are allowed.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—The Transact-SQL **CREATE PROCEDURE** statement is different.
- SQLJ—The syntax extensions for Java result sets are as specified in the proposed SQLJ1 standard.

Permissions

Unless creating a temporary procedure, a user must have the CREATE PROCEDURE system privilege to create a procedure for themselves. To create UDF procedure for others, a user must specify an owner and have either the CREATE ANY PROCEDURES or CREATE ANY OBJECT system privilege. If a procedure has an external reference, a user must also have the CREATE EXTERNAL REFERENCE system privilege, in addition to the previously mentioned system privileges, regardless of whether or not they are the owner of procedure.

CREATE PROCEDURE Statement (Table UDF)

Creates an interface to an external table user-defined function (table UDF). Users must be specifically licensed to use table UDFs.

For **CREATE PROCEDURE** reference information for external procedures, see *CREATE PROCEDURE Statement (External Procedures)*. For **CREATE PROCEDURE** reference information for Java UDFs, see *CREATE PROCEDURE Statement (Java UDF)*.

Syntax

```
CREATE[ OR REPLACE ] PROCEDURE
[ owner.]procedure-name ( [ parameter[, ...]] )
| RESULT result-column [, ...] )
[ SQL SECURITY { INVOKER | DEFINER } ]
EXTERNAL NAME 'external-call'
```

```
parameter:
  [ IN ] parameter-name data-type [ DEFAULT expression ]
  | [ IN ] parameter-name table-type

table-type:
  TABLE( column-name data-type [, ...] )

external-call:
  [column-name:]function-name@library; ...
```

Parameters

- **IN** – the parameter is an object that provides a value for a scalar parameter or a set of values for a TABLE parameter to the UDF.

Note: TABLE parameters cannot be declared as INOUT or OUT. You can only have one table parameter (the position of which is not important).

- **OR REPLACE** – specifying **OR REPLACE (CREATE OR REPLACE PROCEDURE)** creates a new procedure, or replaces an existing procedure with the same name. This clause changes the definition of the procedure, but preserves existing permissions. An error is returned if you attempt to replace a procedure that is already in use.
- **RESULT** – declares the column names and their data types for the result set of the external UDF. The data types of the columns must be a valid SQL data type (e.g., a column in the result set cannot have TABLE as data type). The set of datums in the result implies the TABLE. External UDFs can only have one result set of type TABLE.

Note: TABLE is not an output value. A table UDF cannot have LONG VARBINARY or LONG VARCHAR data types in its result set, but a table parameterized function (TPF) can have large object (LOB) data in its result set.

A TPF cannot produce LOB data, but can have columns in the result set as LOB data types. However, the only way to get LOB data in the output is to pass a column from an input table to the output table. The describe attribute

EXTFNAPIV4_DESCRIBE_COL_VALUES_SUBSET_OF_INPUT allows this, as illustrated in the sample file tpf_blob.cxx.

- **SQL SECURITY** – defines whether the procedure is executed as the INVOKER (the user who is calling the UDF), or as the DEFINER (the user who owns the UDF). The default is DEFINER.

When SQL SECURITY INVOKER is specified, more memory is used because annotation must be done for each user that calls the procedure. Also, when SQL SECURITY INVOKER is specified, name resolution is done as the invoker as well. Therefore, care should be taken to qualify all object names (tables, procedures, and so on) with their appropriate owner. For example, suppose user1 creates this procedure:

```
CREATE PROCEDURE user1.myProcedure ()
  RESULT( columnA INT )
```

```
SQL SECURITY INVOKER
BEGIN
    SELECT columnA FROM table1;
END;
```

If user2 attempts to run this procedure and a table user2.table1 does not exist, a table lookup error results. Additionally, if a user2.table1 does exist, that table is used instead of the intended user1.table1. To prevent this situation, qualify the table reference in the statement (user1.table1, instead of just table1).

- **EXTERNAL NAME** – An external UDF must have EXTERNAL NAME clause which defines an interface to a function written in a programming language such as C. The function is loaded by the database server into its address space.

The library name can include the file extension, which is typically .dll on Windows and .so on UNIX. In the absence of the extension, the software appends the platform-specific default file extension for libraries. This is a formal example.

```
CREATE PROCEDURE mystring( IN instr CHAR(255),
    IN input_table TABLE(A INT) )
    RESULT (CHAR(255))
EXTERNAL NAME
'mystring@mylib.dll;Unix:mystring@mylib.so'
```

A simpler way to write the preceding EXTERNAL NAME clause, using platform-specific defaults, is as follows:

```
CREATE PROCEDURE mystring( IN instr CHAR(255),
    IN input_table TABLE(A INT) )
    RESULT (CHAR(255))
EXTERNAL NAME 'mystring@mylib'
```

Usage

You define table UDFs using the `a_v4_extfn` API. **CREATE PROCEDURE** statement reference information for external procedures that do not use the `a_v3_extfn` or `a_v4_extfn` APIs is located in a separate topic. **CREATE PROCEDURE** statement reference information for Java UDFs is located in a separate topic.

The **CREATE PROCEDURE** statement creates a procedure in the database. To create a procedure for themselves, a user must have the CREATE PROCEDURE system privilege. To create a procedure for others, a user must specify the owner of the procedure and must have either the CREATE ANY PROCEDURE or CREATE ANY OBJECT system privilege. If the procedure contains an external reference, the user must have the CREATE EXTERNAL REFERENCE system privilege in addition to previously mentioned system privileges, regardless of who owns the procedure.

If a stored procedure returns a result set, it cannot also set output parameters or return a return value.

SQL Statements

When referencing a temporary table from multiple procedures, a potential issue can arise if the temporary table definitions are inconsistent and statements referencing the table are cached. Use caution when referencing temporary tables within procedures.

You can use the **CREATE PROCEDURE** statement to create external table UDFs implemented in a different programming language than SQL. However, be aware of the table UDF restrictions before creating external UDFs.

The data type for a scalar parameter, a result column, and a column of a TABLE parameter must be a valid SQL data type.

Parameter names must conform to the rules for other database identifiers such as column names. They must be a valid SQL data type.

TPFs support a mix scalar parameters and single table parameter. A TABLE parameter must define a schema for an input set of rows to be processed by the UDF. The definition of a TABLE parameter includes column names and column data types.

```
TABLE (c1 INT, c2 CHAR(20))
```

The above example defines a schema with the two columns c1 and c2 of types INT and CHAR(20). Each row processed by the UDF must be a tuple with two (2) values. Table parameters, unlike scalar parameters cannot be assigned a default value.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—The Transact-SQL **CREATE PROCEDURE** statement is different.
- SQLJ—The syntax extensions for Java result sets are as specified in the proposed SQLJ1 standard.

Permissions

Unless creating a temporary procedure, a user must have the CREATE PROCEDURE system privilege to create a UDF for themselves. To create a UDF for others, they must specify the owner of the procedure and must have either the CREATE ANY PROCEDURE or CREATE ANY OBJECT system privilege. If the procedure contains an external reference, a user must also have the CREATE EXTERNAL REFERENCE system privilege, in addition to the previously mentioned system privileges.

CREATE ROLE Statement

Creates a new role, extends an existing user to act as a role, or manages role administrators on a role.

Syntax

```
CREATE [ OR REPLACE ] ROLE { role_name | FOR USER userID }
  [ WITH ADMIN [ ONLY ] admin_name [...], [ SYS_MANAGE_ROLES_ROLE ]
```

Parameters

- **role_name** – unless you are using the OR REPLACE clause, *role_name* cannot already exist in the database.
- **OR REPLACE** – *role_name* must already exist in the database. If *role_name* does not already exist, a new user-defined role is created. All current administrators are replaced by those specified in the *admin_name* [...] clause as follows:
 - All existing role administrators granted the WITH ADMIN OPTION not included on the new role administrators list become members of the role with no administrative rights on the role.
 - All existing role administrators granted the WITH ADMIN ONLY OPTION not included on the new role administrators list are removed as members of the role.

When using the OR REPLACE clause, if an existing role administrator is included on the new role administrators list he or she retains his or her original administrative rights if they are higher than the replacement rights. For example, User A is an existing role administrator originally granted WITH ADMIN rights on the role. New role administrators are granted WITH ADMIN ONLY rights. If User A is included on this list, User A retains the higher WITH ADMIN rights.

- **FOR USER** – when using the FOR USER clause without the OR REPLACE, *userID* must be the name of an existing user that currently does not have the ability to act as a role.
- **admin_name** – list of users to be designated administrators of the role.
- **WITH ADMIN** – each *admin_name* specified is granted administrative privileges over the role in addition to all underlying system privileges. WITH ADMIN clause is not valid when SYS_MANAGE_ROLES_ROLE is included on the list.
- **WITH ADMIN ONLY** – each *admin_name* specified is granted administrative privileges only over the role, not the underlying system privileges.
- **SYS_MANAGE_ROLES_ROLE** – allows global role administrators to administer the role. Can be specified in conjunction with the WITH ADMIN ONLY clause.

Examples

- **Example 1** – creates the role `Sales`. Only global role administrator can administer the role.

```
CREATE ROLE Sales
```

- **Example 2** – extends the existing user `Jane` to act as a role.

```
CREATE OR REPLACE ROLE FOR USER Jane
```

- **Example 3** – creates the role `Finance` with `Mary` and `Jeff` as role administrators with administrative rights to the role. Global role administrators cannot administer this role.

```
CREATE ROLE Finance  
WITH ADMIN Mary, Jeff
```

- **Example 3** – creates the role `Marketing` with `Mary` and `Jeff` as role administrators. Global role administrators can also manage this role.

```
CREATE ROLE Finance  
WITH ADMIN ONLY Mary, Jeff, SYS_MANAGE_ROLES_ROLE
```

- **Example 4** – `Finance` is an existing role with `Harry` and `Susan` as role administrators with administrative rights. You want to keep `Susan` as an administrator, replace `Harry`, and add the global role administrator. The new role administrators will have administrative rights only.

This statement keeps `Susan` as an administrator, but `Susan` retains administrative rights to the role since the original administrative rights granted were higher. `Harry` is replaced by `Bob` and `Sarah`, with administrative rights only, and the global role administrator is added to the role. `Harry` remains a member of the role, but has no administrative rights.

```
CREATE OR REPLACE ROLE Finance  
WITH ADMIN ONLY Susan, Bob, Sarah, SYS_MANAGE_ROLE_ROLE
```

Usage

If you specify role administrators (*admin_name*), but do not include the global role administrator (`SYS_MANAGE_ROLES_ROLE`), global role administrators will be unable to manage the new role. Therefore, it is recommended that you not specify role administrators during the creation process. Use the `OR REPLACE` clause to add them afterwards.

If you do not specify an `ADMIN` clause, the default `WITH ADMIN ONLY` clause is used and the default administrator is the global roles administrator (`SYS_MANAGE_ROLES_ROLE`).

When replacing role administrators, if the role has a global role administrator, it must be included on the new role administrators list or it is removed from the role.

However, when using the `WITH ADMIN` clause to grant role administrators, since the clause is not valid for global role administrators, you must use the **GRANT ROLE** statement to re-add the global role administrator (`SYS_MANAGE_RILES_ROLE`) to the role. Failure to perform this grant means global role administrators are unable to manage the role.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

- Create a new role – Requires the MANAGE ROLES system privilege.
- OR REPLACE clause – Requires the MANAGE ROLES system privilege along with administrative rights over the role being replaced.

CREATE SCHEMA Statement

Creates a schema, which is a collection of tables, views, and permissions and their associated permissions, for a database user.

Syntax

```
CREATE SCHEMA AUTHORIZATION userid
    ... [ { create-table-statement
        | create-view-statement
        | grant-statement } ] ...
```

Usage

The *userid* must be the user ID of the current connection. You cannot create a schema for another user. The user ID is not case-sensitive.

If any of the statements in the **CREATE SCHEMA** statement fail, the entire **CREATE SCHEMA** statement is rolled back.

CREATE SCHEMA statement is simply a way to collect individual **CREATE** and **GRANT** statements into one operation. There is no **SCHEMA** database object created in the database, and to drop the objects you must use individual **DROP TABLE** or **DROP VIEW** statements. To revoke permissions, use a **REVOKE** statement for each permission granted.

Note: The **CREATE SCHEMA** statement is invalid on an active multiplex.

Individual **CREATE** or **GRANT** statements are not separated by statement delimiters. The statement delimiter marks the end of the **CREATE SCHEMA** statement itself.

The individual **CREATE** or **GRANT** statements must be ordered such that the objects are created before permissions are granted on them.

Creating more than one schema for a user is not recommended and might not be supported in future releases.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—SAP Sybase IQ does not support the use of **REVOKE** statements within the **CREATE SCHEMA** statement, and does not allow its use within Transact-SQL batches or procedures.

Permissions

Requires the CREATE ANY OBJECT system privilege.

See also

- *CREATE TABLE Statement* on page 205
- *CREATE VIEW Statement* on page 234
- *GRANT CREATE Statement* on page 308

CREATE SEQUENCE statement

Creates a sequence that can be used to generate primary key values that are unique across multiple tables, and for generating default values for a table. This statement applies to SAP Sybase IQ catalog store tables only.

Syntax

```
CREATE [ OR REPLACE ] SEQUENCE [ owner. ] sequence-name  
[ INCREMENT BY signed-integer ]  
[ START WITH signed-integer ]  
[ MINVALUE signed-integer | NO MINVALUE ]  
[ MAXVALUE signed-integer | NO MAXVALUE ]  
[ CACHE integer | NO CACHE ]  
[ CYCLE | NO CYCLE ]
```

Parameters

OR REPLACE clause – Specifying OR REPLACE creates a new sequence, or replaces an existing sequence with the same name. If you do not use the OR REPLACE clause, an error is returned if you specify the name of a sequence that already exists for the current user.

INCREMENT BY clause – Defines the amount the next sequence value is incremented from the last value assigned. The default is 1. Specify a negative value to generate a descending sequence. An error is returned if the INCREMENT BY value is 0.

START WITH clause – Defines the starting sequence value. If you do not specify a value for the START WITH clause, MINVALUE is used for ascending sequences and MAXVALUE is used for descending sequences. An error is returned if the START WITH value is beyond the range specified by MINVALUE or MAXVALUE.

MINVALUE clause – Defines the smallest value generated by the sequence. The default is 1. An error is returned if MINVALUE is greater than $(2^{63}-1)$ or less than $-(2^{63}-1)$. An error is also returned if MINVALUE is greater than MAXVALUE.

MAXVALUE clause – Defines the largest value generated by the sequence. The default is $2^{63}-1$. An error is returned if MAXVALUE is greater than $2^{63}-1$ or less than $-(2^{63}-1)$.

CACHE clause – Specifies the number of preallocated sequence values that are kept in memory for faster access. When the cache is exhausted, the sequence cache is repopulated and a corresponding entry is written to the transaction log. At checkpoint time, the current value of the cache is forwarded to the `ISYSSEQUENCE` system table. The default is 100.

CYCLE clause – Specifies whether values should continue to be generated after the maximum or minimum value is reached.

The default is `NO CYCLE`, which returns an error once the maximum or minimum value is reached.

Remarks

A sequence is a database object that allows the automatic generation of numeric values. A sequence is not bound to a specific or unique table column and is only accessible through the table column to which it is applied.

Sequences can generate values in one of the following ways:

- Increment or decrement monotonically without bound

- Increment or decrement monotonically to a user-defined limit and stop

- Increment or decrement monotonically to a user-defined limit and cycle back to the beginning and start again

You control the behavior when the sequence runs out of values using the `CYCLE` clause.

If a sequence is increasing and it exceeds the `MAXVALUE`, `MINVALUE` is used as the next sequence value if `CYCLE` is specified. If a sequence is decreasing and it falls below `MINVALUE`, `MAXVALUE` is used as the next sequence value if `CYCLE` is specified. If `CYCLE` is not specified, an error is returned.

Sequence values cannot be used with views or materialized view definitions.

Privileges

You must have the `CREATE ANY SEQUENCE` or `CREATE ANY OBJECT` system privilege to create sequences.

Side effects

None

Standards and compatibility

- **SQL/2008** – Sequences comprise SQL/2008 language feature T176. SAP Sybase IQ does not allow optional specification of the sequence data type—this behavior can be achieved with a CAST when using the sequence.

In addition, the following are vendor extensions:

- CACHE clause
- OR REPLACE syntax
- CURRVAL expression
- Use of sequences in DEFAULT expressions

Example

The following example creates a sequence named Test that starts at 4, increments by 2, does not cycle, and caches 15 values at a time:

```
CREATE SEQUENCE Test
START WITH 4
INCREMENT BY 2
NO MAXVALUE
NO CYCLE
CACHE 15;
```

CREATE SERVER Statement

Adds a server to the ISYSSERVER table.

Syntax

```
CREATE SERVER server-name
  CLASS 'server-class'
  USING 'connection-info'
  [ READ ONLY ]
```

server-class:

```
{ ASAJDBC
  | ASEJDBC
  | SAODBC
  | ASEODBC
  | DB2ODBC
  | MSSODBC
  | ORAODBC
  | ODBC }
```

connection-info:

```
{ machine-name:port-number [ /dbname ] | data-source-name }
```

Parameters

- **USING** – if a JDBC-based server class is used, the USING clause is *hostname:port-number [/dbname]* where:
 - **hostname** – the machine on which the remote server runs.
 - **portnumber** – the TCP/IP port number on which the remote server listens. The default port number for SAP Sybase IQ and SQL Anywhere is 2638.
 - **dbname** – for SQL Anywhere remote servers, if you do not specify a *dbname*, the default database is used. For Adaptive Server Enterprise, the default is the master database, and an alternative to using *dbname* is to another database by some other means (for example, in the **FORWARD TO** statement).

If an ODBC-based server class is used, the USING clause is the *data-source-name*, which is the ODBC Data Source Name.

- **READ ONLY** – specifies that the remote server is a read-only data source. Any update request is rejected by SAP Sybase IQ.

Examples

- **Example 1** – create a remote server for the JDBC-based Adaptive Server Enterprise server named `ase_prod`. Its machine name is “banana” and port number is 3025.

```
CREATE SERVER ase_prod
CLASS 'asejdbc'
USING 'banana:3025'
```

- **Example 2** – create a SQL Anywhere remote server named `testasa` on the machine “apple” with listening on port number 2638:

```
CREATE SERVER testasa
CLASS 'asajdbc'
USING 'apple:2638'
```

- **Example 3** – create a remote server for the Oracle server named `oracle723`. Its ODBC Data Source Name is “oracle723”:

```
CREATE SERVER oracle723
CLASS 'oraodbc'
USING 'oracle723'
```

Usage

CREATE SERVER defines a remote server from the SAP Sybase IQ catalogs.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

Requires the SERVER OPERATOR system privilege.

See also

- *ALTER SERVER Statement* on page 41
- *DROP SERVER Statement* on page 269

CREATE SERVICE Statement

Permits a database server to act as a Web server.

Syntax

```
CREATE SERVICE service-name-string
  TYPE service-type-string
  [ attributes ] [
  AS statement ]

attributes:
  [ AUTHORIZATION { ON | OFF } ]
  [ SECURE { ON | OFF } ]
  [ USER { user-name | NULL } ]
  [ URL [ PATH/ ] { ON | OFF | ELEMENTS } ]
  [ USING { SOAP-prefix | NULL } ]

service-type-string:
  { 'RAW '
  | 'HTML '
  | 'XML '
  | 'SOAP '
  | 'DISH ' }
```

Parameters

- **service-name-string** – Web service names may be any sequence of alphanumeric characters or “/”, “-”, “_”, “.”, “!”, “~”, “*”, “””, “(”, or “)”, except that the first character cannot begin with a slash (/) and the name cannot contain two or more consecutive slash characters.
- **AUTHORIZATION** – determines whether users must specify a user name and password when connecting to the service. The default value is ON.

- If authorization is **OFF**, the **AS** clause is required and a single user must be identified by the **USER** clause. All requests are run using that user's account and permissions.
- If authorization is **ON**, all users must provide a user name and password. Optionally, you can limit the users that are permitted to use the service by providing a user or role name using the **USER** clause. If the user name is **NULL**, all known users can access the service.

Run production systems with authorization turned on. Grant permission to use the service by adding users to a role.

- **SECURE** – indicates whether unsecure connections are accepted. **ON** indicates that only **HTTPS** connections are to be accepted. Service requests received on the **HTTP** port are automatically redirected to the **HTTPS** port. If set to **OFF**, both **HTTP** and **HTTPS** connections are accepted. The default value is **OFF**.
- **USER** – if authorization is disabled, this parameter becomes mandatory and specifies the user ID used to execute all service requests. If authorization is enabled (the default), this optional clause identifies the user or role permitted access to the service. The default value is **NULL**, which grants access to all users.
- **URL** – determines whether URI paths are accepted and, if so, how they are processed. **OFF** indicates that nothing must follow the service name in a URI request. **ON** indicates that the remainder of the URI is interpreted as the value of a variable named *url*. **ELEMENTS** indicates that the remainder of the URI path is to be split at the slash characters into a list of up to 10 elements. The values are assigned to variables named *url* plus a numeric suffix of between 1 and 10; for example, the first three variable names are *url1*, *url2*, and *url3*. If fewer than 10 values are supplied, the remaining variables are set to **NULL**. If the service name ends with the character */*, then **URL** must be set to **OFF**. The default value is **OFF**.
- **USING** – applies only to **DISH** services. The parameter specifies a name prefix. Only **SOAP** services whose names begin with this prefix are handled.
- **service-type-string** – identifies the type of the service. The type must be one of the listed service types. There is no default value.
 - **RAW** – sends the result set to the client without any further formatting. You can produce formatted documents by generating the required tags explicitly within your procedure.
 - **HTML** – formats the result set of a statement or procedure into an **HTML** document that contains a table.
 - **XML** – assumes the result set is an **XML** format. If it is not already so, it is automatically converted to **XML RAW** format.
 - **SOAP** – formats the result set as a Simple Object Access Protocol (**SOAP**) response. The request must be a valid **SOAP** request. For more information about the **SOAP** standards, see www.w3.org/TR/SOAP
 - **DISH** – determine **SOAP** Handler, or **DISH**, service acts as a proxy for one or more **SOAP** services. In use, it acts as a container that holds and provides access to a number

SQL Statements

of SOAP services. A Web Services Description Language (WSDL) file is automatically generated for each of the included SOAP services. The included SOAP services are identified by a common prefix, which must be specified in the **USING** clause.

- **statement** – if the statement is NULL, the URI must specify the statement to be executed. Otherwise, the specified SQL statement is the only one that can be executed through the service. The statement is mandatory for SOAP services, and ignored for DISH services. The default value is NULL.

All services that are run in production systems must define a statement. The statement can be NULL only if authorization is enabled.

Examples

- **Example 1** – sets up a Web server quickly, start a database server with the `-xs` switch, then execute this statement:

```
CREATE SERVICE tables TYPE 'HTML'  
AUTHORIZATION OFF USER DBA  
AS SELECT * FROM SYS.ISYSTAB
```

After executing this statement, use any Web browser to open the URL `http://localhost/tables`.

Usage

The **CREATE SERVICE** statement causes the database server to act as a web server. A new entry is created in the `SYSWEBSERVICE` system table.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the `MANAGE ANY WEB SERVICE` system privilege.

See also

- *ALTER SERVICE Statement* on page 44
- *DROP SERVICE Statement* on page 269

CREATE SPATIAL REFERENCE SYSTEM Statement

Creates or replaces a spatial reference system.

Syntax

```

{ CREATE [ OR REPLACE ] SPATIAL REFERENCE SYSTEM
  | CREATE SPATIAL REFERENCE SYSTEM IF NOT EXISTS }
  srs-name
  [ srs-attribute ] [ srs-attribute ... ]

srs-attribute:
  SRID srs-id
  | DEFINITION { definition-string | NULL }
  | ORGANIZATION { organization-name IDENTIFIED BY organization-srs-id
  | NULL }
  | TRANSFORM DEFINITION { transform-definition-string | NULL }
  | LINEAR UNIT OF MEASURE linear-unit-name
  | ANGULAR UNIT OF MEASURE { angular-unit-name | NULL }
  | TYPE { ROUND EARTH | PLANAR }
  | COORDINATE coordinate-name { UNBOUNDED | BETWEEN low-number
AND high-number }
  | ELLIPSOID SEMI MAJOR AXIS semi-major-axis-length { SEMI MINOR AXIS
semi-minor-axis-length
  | INVERSE FLATTENING inverse-flattening-ratio }
  | SNAP TO GRID { grid-size | DEFAULT }
  | TOLERANCE { tolerance-distance | DEFAULT }
  | POLYGON FORMAT polygon-format
  | STORAGE FORMAT storage-format

grid-size:
  DOUBLE : usually between 0 and 1

axis-order:
  { 'x/y/z/m' | 'long/lat/z/m' | 'lat/long/z/m' }

polygon-format:
  { 'CounterClockWise' | 'Clockwise' | 'EvenOdd' }

storage-format:
  { 'Internal' | 'Original' | 'Mixed' }

```

Parameters

- **OR REPLACE** – Specifying OR REPLACE creates the spatial reference system if it does not already exist in the database, and replaces it if it does exist. An error is returned if you attempt to replace a spatial reference system while it is in use. An error is also returned if you attempt to replace a spatial reference system that already exists in the database without specifying the OR REPLACE clause.

- **IF NOT EXISTS** – Specifying CREATE SPATIAL REFERENCE IF NOT EXISTS checks to see if a spatial reference system by that name already exists. If it does not exist, the database server creates the spatial reference system. If it does exist, no further action is performed and no error is returned.
- **IDENTIFIED BY** – the SRID (*srs-id*) for the spatial reference system. If the spatial reference system is defined by an organization with an *organization-srs-id*, then *srs-id* should be set to that value.

If the IDENTIFIED BY clause is not specified, then the SRID defaults to the *organization-srs-id* defined by either the ORGANIZATION clause or the DEFINITION clause. If neither clause defines an *organization-srs-id* that could be used as a default SRID, an error is returned.

When the spatial reference system is based on a well known coordinate system, but has a different geodesic interpretation, set the srs-id value to be 1000000000 (one billion) plus the well known value. For example, the SRID for a planar interpretation of the geodetic spatial reference system WGS 84 (ID 4326) would be 1000004326.

With the exception of SRID 0, spatial reference systems provided by SAP Sybase IQ that are not based on well known systems are given a SRID of 2000000000 (two billion) and above. The range of SRID values from 2000000000 to 2147483647 is reserved by SAP Sybase IQ and you should not create SRIDs in this range.

To reduce the possibility of choosing a SRID that is reserved by a defining authority such as OGC or by other vendors, you should not choose a SRID in the range 0 - 32767 (reserved by EPSG), or in the range 2147483547 - 2147483647.

Also, since the SRID is stored as a signed 32-bit integer, the number cannot exceed 231-1 or 2147483647.

- **DEFINITION** – set, or override, default coordinate system settings. If any attribute is set in a clause other than the DEFINITION clause, it takes the value specified in the other clause regardless of what is specified in the DEFINITION clause.

definition-string is a string in the Spatial Reference System Well Known Text syntax as defined by SQL/MM and OGC. For example, the following query returns the definition for WGS 84.

```
SELECT ST_SpatialRefSys::ST_FormatWKT( definition )
FROM ST_SPATIAL_REFERENCE_SYSTEMS
WHERE srs_id=4326;
```

In Interactive SQL, if you double-click the value returned, an easier to read version of the value appears.

When the DEFINITION clause is specified, definition-string is parsed and used to choose default values for attributes. For example, definition-string may contain an AUTHORITY element that defines the organization-name and *organization-srs-id*.

Parameter values in definition-string are overridden by values explicitly set using the SQL statement clauses. For example, if the ORGANIZATION clause is specified, it overrides the value for ORGANIZATION in definition-string.

- **ORGANIZATION** – information about the organization that created the spatial reference system that the spatial reference system is based on.
- **TRANSFORM DEFINITION** – a description of the transform to use for the spatial reference system. Currently, only the PROJ.4 transform is supported. The transform definition is used by the ST_Transform method when transforming data between spatial reference systems. Some transforms may still be possible even if there is no transform-definition-string defined.
- **LINEAR UNIT OF MEASURE** – the linear unit of measure for the spatial reference system. The value you specify must match a linear unit of measure defined in the ST_UNITS_OF_MEASURE system view.

If this clause is not specified, and is not defined in the DEFINITION clause, the default is METRE. To add predefined units of measure to the database, use the sa_install_feature system procedure.

To add custom units of measure to the database, use the CREATE SPATIAL UNIT OF MEASURE statement.

Note: While both METRE and METER are accepted spellings, METRE is preferred as it conforms to the SQL/MM standard.

- **ANGULAR UNIT OF MEASURE** – the angular unit of measure for the spatial reference system. The value you specify must match an angular unit of measure defined in the ST_UNITS_OF_MEASURE system table.

If this clause is not specified, and is not defined in the DEFINITION clause, the default is DEGREE for geographic spatial reference systems and NULL for non-geographic spatial reference systems.

The angular unit of measure must be non-NULL for geographic spatial reference systems and it must be NULL for non-geographic spatial reference systems.

The angular unit of measure must be non-NULL for geographic spatial reference systems and it must be NULL for non-geographic spatial reference systems. To add predefined units of measure to the database, use the sa_install_feature system procedure.

To add custom units of measure to the database, use the CREATE SPATIAL UNIT OF MEASURE statement.

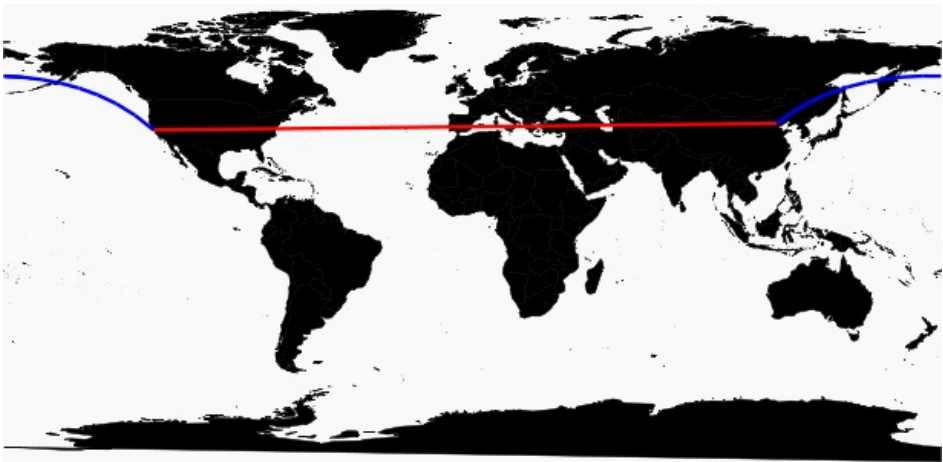
- **TYPE** – control how the SRS interprets lines between points. For geographic spatial reference systems, the TYPE clause can specify either ROUND EARTH (the default) or PLANAR. The ROUND EARTH model interprets lines between points as great elliptic arcs. Given two points on the surface of the Earth, a plane is selected that intersects the two

points and the center of the Earth. This plane intersects the Earth, and the line between the two points is the shortest distance along this intersection.

For two points that lie directly opposite each other, there is not a single unique plane that intersects the two points and the center of the Earth. Line segments connecting these antipodal points are not valid and give an error in the ROUND EARTH model.

The ROUND EARTH model treats the Earth as a spheroid and selects lines that follow the curvature of the Earth. In some cases, it may be necessary to use a planar model where a line between two points is interpreted as a straight line in the equirectangular projection where $x=\text{long}$, $y=\text{lat}$.

In the following example, the blue line shows the line interpretation used in the ROUND EARTH model and the red line shows the corresponding PLANAR model.



The PLANAR model may be used to match the interpretation used by other products. The PLANAR model may also be useful because there are some limitations for methods that are not supported in the ROUND EARTH model (such as `ST_Area`, `ST_ConvexHull`) and some are partially supported (`ST_Distance` only supported between point geometries). Geometries based on circularstrings are not supported in ROUND EARTH spatial reference systems.

For non-geographic SRSSs, the type must be PLANAR (and that is the default if the TYPE clause is not specified and either the DEFINITION clause is not specified or it uses a non-geographic definition).

- **COORDINATE** – the bounds on the spatial reference system's dimensions. coordinate-name is the name of the coordinate system used by the spatial reference system. For non-geographic coordinate systems, coordinate-name can be x, y, or m. For geographic coordinate systems, coordinate-name can be LATITUDE, LONGITUDE, z, or m.

Specify UNBOUNDED to place no bounds on the dimensions. Use the BETWEEN clause to set low and high bounds.

The X and Y coordinates must have associated bounds. For geographic spatial reference systems, the longitude coordinate is bounded between -180 and 180 degrees and the latitude coordinate is bounded between -90 and 90 degrees by default unless the COORDINATE clause overrides these settings. For non-geographic spatial reference systems, the CREATE statement must specify bounds for both X and Y coordinates.

LATITUDE and LONGITUDE are used for geographic coordinate systems. The bounds for LATITUDE and LONGITUDE default to the entire Earth, if not specified.

- **ELLIPSOID** – the values to use for representing the Earth as an ellipsoid for spatial reference systems of type ROUND EARTH. If the DEFINITION clause is present, it can specify ellipsoid definition. If the ELLIPSOID clause is specified, it overrides this default ellipsoid.

The Earth is not a perfect sphere because the rotation of the Earth causes a flattening so that the distance from the center of the Earth to the North or South pole is less than the distance from the center to the equator. For this reason, the Earth is modeled as an ellipsoid with different values for the semi-major axis (distance from center to equator) and semi-minor axis (distance from center to the pole). It is most common to define an ellipsoid using the semi-major axis and the inverse flattening, but it can instead be specified using the semi-minor axis (for example, this approach must be used when a perfect sphere is used to approximate the Earth). The semi-major and semi-minor axes are defined in the linear units of the spatial reference system, and the inverse flattening (1/f) is a ratio:

$$1/f = (\text{semi-major-axis}) / (\text{semi-major-axis} - \text{semi-minor-axis})$$

product-name uses the ellipsoid definition when computing distance in geographic spatial reference systems.

- **SNAP TO GRID** – flat-Earth (planar) spatial reference systems, use the SNAP TO GRID clause to define the size of the grid SAP Sybase IQ uses when performing calculations. By default, SAP Sybase IQ selects a grid size so that 12 significant digits can be stored at all points in the space bounds for X and Y. For example, if a spatial reference system bounds X between -180 and 180 and Y between -90 and 90, then a grid size of 0.000000001 (1E-9) is selected.
- **TOLERANCE** – flat-Earth (planar) spatial reference systems, use the TOLERANCE clause to specify the precision to use when comparing points. If the distance between two points is less than tolerance-distance, the two points are considered equal. Setting tolerance-distance allows you to control the tolerance for imprecision in the input data or limited internal precision. By default, tolerance-distance is set to be equal to grid-size.

When set to 0, two points must be exactly equal to be considered equal.

For round-Earth spatial reference systems, TOLERANCE must be set to 0.

- **POLYGON FORMAT** – internally, SAP Sybase IQ interprets polygons by looking at the orientation of the constituent rings. As one travels a ring in the order of the defined points, the inside of the polygon is on the left side of the ring. The same rules are applied in PLANAR and ROUND EARTH spatial reference systems.

The interpretation used by SAP Sybase IQ is a common but not universal interpretation. Some products use the exact opposite orientation, and some products do not rely on ring orientation to interpret polygons. The POLYGON FORMAT clause can be used to select a polygon interpretation that matches the input data, as needed. The following values are supported:

- **CounterClockwise** – input follows SAP Sybase IQ's internal interpretation: the inside of the polygon is on the left side while following ring orientation.
- **Clockwise** – input follows the opposite of SAP Sybase IQ's approach: the inside of the polygon is on the right side while following ring orientation.
- **EvenOdd** – (default) The orientation of rings is ignored and the inside of the polygon is instead determined by looking at the nesting of the rings, with the exterior ring being the largest ring and interior rings being smaller rings inside this ring. A ray is traced from a point within the rings and radiating outward crossing all rings. If the number the ring being crossed is an even number, it is an outer ring. If it is odd, it is an inner ring.
- **STORAGE FORMAT** – control what is stored when spatial data is loaded into the database. Possible values are:
 - **Internal** – SAP Sybase IQ stores only the normalized representation. Specify this when the original input characteristics do not need to be reproduced. This is the default for planar spatial reference systems (TYPE PLANAR).
 - **Original** – SAP Sybase IQ stores only the original representation. The original input characteristics can be reproduced, but all operations on the stored values must repeat normalization steps, possibly slowing down operations on the data.
 - **Mixed** – SAP Sybase IQ stores the internal version and, if it is different from the original version, SQL Anywhere stores the original version as well. By storing both versions, the original representation characteristics can be reproduced and operations on stored values do not need to repeat normalization steps. However, storage requirements may increase significantly because potentially two representations are being stored for each geometry. Mixed is the default format for round-Earth spatial reference systems (TYPE ROUND EARTH).

Examples

- **Example 1** – creates a spatial reference system named mySpatialRS:

```
CREATE SPATIAL REFERENCE SYSTEM "mySpatialRS"  
IDENTIFIED BY 1000026980  
LINEAR UNIT OF MEASURE "metre"  
TYPE PLANAR  
COORDINATE X BETWEEN 171266.736269555 AND 831044.757769222  
COORDINATE Y BETWEEN 524881.608973277 AND 691571.125115319  
DEFINITION 'PROJCS["NAD83 / Kentucky South",  
GEOGCS["NAD83",  
DATUM["North American Datum 1983",  
SPHEROID["GRS 1980",  
6378137,298.257222101,AUTHORITY["EPSG","7019"]],  
AUTHORITY["EPSG","6269"]],
```

```

PRIMEM["Greenwich",0,AUTHORITY["EPSG","8901"]],
UNIT["degree",0.01745329251994328,AUTHORITY["EPSG","9122"]],
AUTHORITY["EPSG","4269"]],
UNIT["metre",1,AUTHORITY["EPSG","9001"]],
PROJECTION["Lambert_Conformal_Conic_2SP"],
PARAMETER["standard_parallel_1",37.93333333333333],
PARAMETER["standard_parallel_2",36.73333333333333],
PARAMETER["latitude_of_origin",36.33333333333334],
PARAMETER["central_meridian",-85.75],
PARAMETER["false_easting",500000],
PARAMETER["false_northing",500000],
AUTHORITY["EPSG","26980"],
AXIS["X",EAST],
AXIS["Y",NORTH]] '
TRANSFORM DEFINITION '+proj=lcc
+lat_1=37.93333333333333+lat_2=36.73333333333333+lat_0=36.3333333
3333334+lon_0=-85.75+x_0=500000+y_0=500000+ellps=GRS80+datum=NAD8
3+units=m+nö_defs';

```

Usage

For a geographic spatial reference system, you can specify both a LINEAR and an ANGULAR unit of measure; otherwise for non-geographic, you specify only a LINEAR unit of measure. The LINEAR unit of measure is used for computing distance between points and areas. The ANGULAR unit of measure tells how the angular latitude/longitude are interpreted and is NULL for projected coordinate systems, non-NULL for geographic coordinate systems.

All derived geometries returned by operations are normalized.

When working with data that is being synchronized with a non-SQL Anywhere database, STORAGE FORMAT should be set to either 'Original' or 'Mixed' so that the original characteristics of the data can be preserved.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- MANAGE ANY SPATIAL OBJECT system privilege.
- CREATE ANY OBJECT system privilege.

CREATE SPATIAL UNIT OF MEASURE Statement

Creates or replaces a spatial unit of measurement.

Syntax

```
CREATE [ OR REPLACE ] SPATIAL UNIT OF MEASURE identifier
  TYPE { LINEAR | ANGULAR }
  [ CONVERT USING number ]
```

Parameters

- **OR REPLACE** – including the OR REPLACE creates a new spatial unit of measure, or replaces an existing spatial unit of measure with the same name. This clause preserves existing privileges. An error is returned if you attempt to replace a spatial unit that is already in use.
- **TYPE** – defines whether the unit of measure is used for angles (ANGULAR) or distances (LINEAR).
- **CONVERT USING** – the conversion factor for the spatial unit relative to the base unit. For linear units, the base unit is METRE. For angular units, the base unit is RADIAN.

Examples

- **Example 1** – creates a spatial unit of measure named Test:

```
CREATE SPATIAL UNIT OF MEASURE Test
  TYPE LINEAR
  CONVERT USING 15;
```

Usage

The CONVERT USING clause is used to define how to convert a measurement in the defined unit of measure to the base unit of measure (radians or meters). The measurement is multiplied by the supplied conversion factor to get a value in the base unit of measure. For example, a measurement of 512 millimeters would be multiplied by a conversion factor of 0.001 to get a measurement of 0.512 metres.

Spatial reference systems always include a linear unit of measure to be used when calculating distances (ST_Distance or ST_Length), or area. For example, if the linear unit of measure for a spatial reference system is miles, then the area unit used is square miles. In some cases, spatial methods accept an optional parameter that specifies the linear unit of measure to use. For example, if the linear unit of measure for a spatial reference system is in miles, you could retrieve the distance between two geometries in meters by using the optional parameter 'metre'.

For projected coordinate systems, the X and Y coordinates are specified in the linear unit of the spatial reference system. For geographic coordinate systems, the latitude and longitude are

specified in the angular units of measure associated with the spatial reference system. In many cases, this angular unit of measure is degrees but any valid angular unit of measure can be used.

You can use the `sa_install_feature` system procedure to add predefined units of measure to your database.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- `MANAGE ANY SPATIAL OBJECT` system privilege.
- `CREATE ANY OBJECT` system privilege.

CREATE TABLE Statement

Creates a new table in the database or on a remote server.

Syntax

```

CREATE [ { GLOBAL | LOCAL } TEMPORARY ] TABLE
  [ IF NOT EXISTS ] [ owner. ] table-name
  ... ( column-definition [ column-constraint ] ...
  [ , column-definition [ column-constraint ] ... ]
  [ , table-constraint ] ... )
  | { ENABLE | DISABLE } RLV STORE

  ... [ IN dbspace-name ]
  ... [ ON COMMIT { DELETE | PRESERVE } ROWS ]
  [ AT location-string ]
  [ PARTITION BY
    range-partitioning-scheme
    | hash-partitioning-scheme
    | composite-partitioning-scheme ]

column-definition:
  column-name data-type
  [ [ NOT ] NULL ]
  [ DEFAULT default-value | IDENTITY ]
  [ PARTITION | SUBPARTITION ( partition-name IN dbspace-name
  [ , ... ] ) ]

default value:
  special-value
  | string
  | global variable
  | [ - ] number
  | ( constant-expression )

```

SQL Statements

```
| built-in-function( constant-expression )
| AUTOINCREMENT
| CURRENT DATABASE
| CURRENT REMOTE USER
| NULL
| TIMESTAMP
| LAST USER

special value:
CURRENT
{ DATE
| TIME
| TIMESTAMP
| USER
| PUBLISHER }
| USER

column-constraint:
[ CONSTRAINT constraint-name ] {
  { UNIQUE
    | PRIMARY KEY
    | REFERENCES table-name [ ( column-name ) ] [ action ]
  }
  [ IN dbspace-name ]
  | CHECK ( condition )
  | IQ UNIQUE ( integer )
}

table-constraint:
[ CONSTRAINT constraint-name ]
{ { UNIQUE ( column-name [ , column-name ] ... )
  | PRIMARY KEY ( column-name [ , column-name ] ... )
}
[ IN dbspace-name ]
| foreign-key-constraint
| CHECK ( condition )
| IQ UNIQUE ( integer )
}

foreign-key-constraint:
FOREIGN KEY [ role-name ] [ ( column-name [ , column-name ] ... ) ]
...REFERENCES table-name [ ( column-name [ , column-name ] ... ) ]
...[ actions ] [ IN dbspace-name ]

actions:
[ ON { UPDATE | DELETE } RESTRICT ]

location-string:
{ remote-server-name. [ db-name ].[ owner ].object-name
  | remote-server-name; [ db-name ];[ owner ];object-name }

range-partitioning-scheme:
RANGE( partition-key ) ( range-partition-decl [,range-partition-decl ... ] )

partition-key:
```



```

    column-name

range-partition-decl:
    VALUES <= ( {constant-expr
        | MAX } [ , { constant-expr
        | MAX } ]... )
    [ IN dbspace-name ]

hash-partitioning-scheme:
    HASH ( partition-key [ , partition-key, ... ] )

composite-partitioning-scheme:
    hash-partitioning-scheme SUBPARTITION range-partitioning-scheme

```

Parameters

- **IN** – used in the column-definition, column-constraint, table-constraint, foreign-key, and partition-decl clauses to specify the dbspace where the object is to be created. If the IN clause is omitted, SAP Sybase IQ creates the object in the dbspace where the table is assigned.

Specify **SYSTEM** with this clause to put either a permanent or temporary table in the catalog store. Specify **IQ_SYSTEM_TEMP** to store temporary user objects (tables, partitions, or table indexes) in **IQ_SYSTEM_TEMP** or, if the **TEMP_DATA_IN_SHARED_TEMP** option is set 'ON', and the **IQ_SHARED_TEMP** dbspace contains RW files, in **IQ_SHARED_TEMP**. (You cannot specify the IN clause with **IQ_SHARED_TEMP**.) All other use of the IN clause is ignored. By default, all permanent tables are placed in the main IQ store, and all temporary tables are placed in the temporary IQ store. Global temporary and local temporary tables can never be in the IQ store.

The following syntax is unsupported:

```
CREATE LOCAL TEMPORARY TABLE tab1(c1 int) IN IQ_SHARED_TEMP
```

- **ON COMMIT** – allowed for temporary tables only. By default, the rows of a temporary table are deleted on **COMMIT**.
- **AT** – creates a proxy table that maps to a remote location specified by the location-string clause. Proxy table names must be 30 characters or less. The **AT** clause supports semicolon (;) delimiters. If a semicolon is present anywhere in the location-string clause, the semicolon is the field delimiter. If no semicolon is present, a period is the field delimiter. This allows file names and extensions to be used in the database and owner fields.

Semicolon field delimiters are used primarily with server classes not currently supported; however, you can also use them in situations where a period would also work as a field delimiter. For example, this statement maps the table `proxy_a` to the SQL Anywhere database `mydb` on the remote server `myasa`:

```
CREATE TABLE proxy_a1
AT 'myasa;mydb;;a1'
```

Foreign-key definitions are ignored on remote tables. Foreign-key definitions on local tables that refer to remote tables are also ignored. Primary key definitions are sent to the remote server if the server supports primary keys.

In a simplex environment, you cannot create a proxy table that refers to a remote table on the same node. In a multiplex environment, you cannot create a proxy table that refers to the remote table defined within the multiplex. .

- **IF NOT EXISTS** – if the named object already exists, no changes are made and an error is not returned.
- **{ ENABLE | DISABLE } RLV STORE** – registers this table with the RLV store for real-time in-memory updates. Not supported for IQ temporary tables. This value overrides the value of the database option **BASE_TABLES_IN_RLV**. Requires the CREATE TABLE system privilege and CREATE permissions on the RLV store dbspace to set this value to ENABLE.
- **column-definition** – defines a table column. Allowable data types are described in *Reference: Building Blocks, Tables, and Procedures >SQL Data Types*. Two columns in the same table cannot have the same name. You can create up to 45,000 columns; however, there might be performance penalties in tables with more than 10,000 columns.
 - **[NOT] NULL**] – includes or excludes NULL values. If NOT NULL is specified, or if the column is in a UNIQUE or PRIMARY KEY constraint, the column cannot contain any NULL values. The limit on the number of columns per table that allow NULLs is approximately $8 * (\text{database-page-size} - 30)$.
 - **DEFAULT default-value** – specify a default column value with the DEFAULT keyword in the CREATE TABLE (and ALTER TABLE) statement. A DEFAULT value is used as the value of the column in any INSERT (or LOAD) statement that does not specify a column value.
 - **DEFAULT AUTOINCREMENT** – the value of the DEFAULT AUTOINCREMENT column uniquely identifies every row in a table. Columns of this type are also known as IDENTITY columns, for compatibility with Adaptive Server Enterprise. The IDENTITY/DEFAULT AUTOINCREMENT column stores sequential numbers that are automatically generated during inserts and updates. When using IDENTITY or DEFAULT AUTOINCREMENT, the column must be one of the integer data types, or an exact numeric type, with scale 0. The column value might also be NULL. You must qualify the specified table name with the owner name.

ON inserts into the table. If a value is not specified for the IDENTITY/DEFAULT AUTOINCREMENT column, a unique value larger than any other value in the column is generated. If an INSERT specifies a value for the column, it is used; if the specified value is not larger than the current maximum value for the column, that value is used as a starting point for subsequent inserts.

Deleting rows does not decrement the IDENTITY/AUTOINCREMENT counter. Gaps created by deleting rows can only be filled by explicit assignment when using an

insert. The database option `IDENTITY_INSERT` must be set to the table name to perform an insert into an `IDENTITY/AUTOINCREMENT` column.

For example, this creates a table with an `IDENTITY` column and explicitly adds some data to it:

```
CREATE TABLE mytable(c1 INT IDENTITY);
SET TEMPORARY OPTION IDENTITY_INSERT = "DBA".mytable;
INSERT INTO mytable VALUES(5);
```

After an explicit insert of a row number less than the maximum, subsequent rows without explicit assignment are still automatically incremented with a value of one greater than the previous maximum.

You can find the most recently inserted value of the column by inspecting the `@@identity` global variable.

- **IDENTITY** – a Transact-SQL-compatible alternative to using the `AUTOINCREMENT` default. In SAP Sybase IQ, the identity column may be created using either the `IDENTITY` or the `DEFAULT AUTOINCREMENT` clause.
- **table-constraint** – helps ensure the integrity of data in the database. There are four types of integrity constraints:
 - **UNIQUE** – identifies one or more columns that uniquely identify each row in the table. No two rows in the table can have the same values in all the named columns. A table may have more than one unique constraint.
 - **PRIMARY KEY** – the same as a `UNIQUE` constraint except that a table can have only one primary-key constraint. You cannot specify the `PRIMARY KEY` and `UNIQUE` constraints for the same column. The primary key usually identifies the best identifier for a row. For example, the customer number might be the primary key for the customer table.
 - **FOREIGN KEY** – restricts the values for a set of columns to match the values in a primary key or uniqueness constraint of another table. For example, a foreign-key constraint could be used to ensure that a customer number in an invoice table corresponds to a customer number in the customer table.

You cannot create foreign-key constraints on local temporary tables. Global temporary tables must be created with `ON COMMIT PRESERVE ROWS`.

- **CHECK** – allows arbitrary conditions to be verified. For example, a check constraint could be used to ensure that a column called `Gender` contains only the values `male` or `female`. No row in a table is allowed to violate a constraint. If an **INSERT** or **UPDATE** statement would cause a row to violate a constraint, the operation is not permitted and the effects of the statement are undone.

Column identifiers in column check constraints that start with the symbol '@' are placeholders for the actual column name. A statement of the form:

```
CREATE TABLE t1(c1 INTEGER CHECK (@foo < 5))
```

is exactly the same as this statement:

```
CREATE TABLE t1(c1 INTEGER CHECK (c1 < 5))
```

Column identifiers appearing in table check constraints that start with the symbol '@' are not placeholders.

If a statement would cause changes to the database that violate an integrity constraint, the statement is effectively not executed and an error is reported. (Effectively means that any changes made by the statement before the error was detected are undone.)

SAP Sybase IQ enforces single-column UNIQUE constraints by creating an HG index for that column.

Note: You cannot define a column with a BIT data type as a UNIQUE or PRIMARY KEY constraint. Also, the default for columns of BIT data type is to not allow NULL values; you can change this by explicitly defining the column as allowing NULL values.

- **column-constraint** – restricts the values the column can hold. Column and table constraints help ensure the integrity of data in the database. If a statement would cause a violation of a constraint, execution of the statement does not complete, any changes made by the statement before error detection are undone, and an error is reported. Column constraints are abbreviations for the corresponding table constraints. For example, these are equivalent:

```
CREATE TABLE Products (
    product_num integer UNIQUE
)
CREATE TABLE Products (
    product_num integer,
    UNIQUE ( product_num )
)
```

Column constraints are normally used unless the constraint references more than one column in the table. In these cases, a table constraint must be used.

- **IQ UNIQUE** – defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit FP. An IQ UNIQUE(n) value explicitly set to 0 loads the column as Flat FP. Columns without an IQ UNIQUE constraint implicitly load as NBit up to the limits defined by the FP_NBIT_AUTOSIZE_LIMIT, FP_NBIT_LOOKUP_MB, and FP_NBIT_ROLLOVER_MAX_MB options:
 - FP_NBIT_AUTOSIZE_LIMIT limits the number of distinct values that load as NBit
 - FP_NBIT_LOOKUP_MB sets a threshold for the total NBit dictionary size
 - FP_NBIT_ROLLOVER_MAX_MB sets the dictionary size for implicit NBit rollovers from NBit to Flat FP
 - FP_NBIT_ENFORCE_LIMITS enforces NBit dictionary sizing limits. This option is OFF by default

Using IQ UNIQUE with an n value less than the FP_NBIT_AUTOSIZE_LIMIT is not necessary. Auto-size functionality automatically sizes all low or medium cardinality

columns as NBit. Use IQ UNIQUE in cases where you want to load the column as Flat FP or when you want to load a column as NBit when the number of distinct values exceeds the FP_NBIT_AUTOSIZE_LIMIT.

Note:

- Consider memory usage when specifying high IQ UNIQUE values. If machine resources are limited, avoid loads with FP_NBIT_ENFORCE_LIMITS='OFF' (default).
Prior to SAP Sybase IQ 16.0, an IQ UNIQUE *n* value > 16777216 would rollover to Flat FP. In 16.0, larger IQ UNIQUE values are supported for tokenization, but may require significant memory resource requirements depending on cardinality and column width.
- BIT, BLOB, and CLOB datatypes do not support NBit dictionary compression. If FP_NBIT_IQ15_COMPATIBILITY='OFF', a non-zero IQ UNIQUE column specification in a CREATE TABLE or ALTER TABLE statement that includes these data types returns an error.

-
- **column-constraint and table-constraint clauses** – column and table constraints help ensure the integrity of data in the database.
 - **PRIMARY KEY or PRIMARY KEY (column-name, ...)** – the primary key for the table consists of the listed columns, and none of the named columns can contain any NULL values. SAP Sybase IQ ensures that each row in the table has a unique primary key value. A table can have only one PRIMARY KEY.
When the second form is used (PRIMARY KEY followed by a list of columns), the primary key is created including the columns in the order in which they are defined, not the order in which they are listed.
When a column is designated as PRIMARY KEY, FOREIGN KEY, or UNIQUE, SAP Sybase IQ creates a High_Group index for it automatically. For multicolumn primary keys, this index is on the primary key, not the individual columns. For best performance, you should also index each column with a HG or LF index separately.
 - **REFERENCES primary-table-name [(primary-column-name)]** – defines the column as a foreign key for a primary key or a unique constraint of a primary table. Normally, a foreign key would be for a primary key rather than an unique constraint. If a primary column name is specified, it must match a column in the primary table which is subject to a unique constraint or primary key constraint, and that constraint must consist of only that one column. Otherwise the foreign key references the primary key of the second table. Primary key and foreign key must have the same data type and the same precision, scale, and sign. Only a non unique single-column HG index is created for a single-column foreign key. For a multicolumn foreign key, SAP Sybase IQ creates a non unique composite HG index. The maximum width of a multicolumn composite key for a unique or non unique HG index is 1KB.

A temporary table cannot have a foreign key that references a base table and a base table cannot have a foreign key that references a temporary table. Local temporary tables cannot have or be referenced by a foreign key.

- **FOREIGN KEY [role-name] [(...)] REFERENCES primary-table-name [(...)]** – defines foreign-key references to a primary key or a unique constraint in another table. Normally, a foreign key would be for a primary key rather than an unique constraint. (In this description, this other table is called the primary table.)

If the primary table column names are not specified, the primary table columns are the columns in the table's primary key. If foreign key column names are not specified, the foreign-key columns have the same names as the columns in the primary table. If foreign-key column names are specified, then the primary key column names must be specified, and the column names are paired according to position in the lists.

If the primary table is not the same as the foreign-key table, either the unique or primary key constraint must have been defined on the referenced key. Both referenced key and foreign key must have the same number of columns, of identical data type with the same sign, precision, and scale.

The value of the row's foreign key must appear as a candidate key value in one of the primary table's rows unless one or more of the columns in the foreign key contains nulls in a null allows foreign key column.

Any foreign-key column not explicitly defined is automatically created with the same data type as the corresponding column in the primary table. These automatically created columns cannot be part of the primary key of the foreign table. Thus, a column used in both a primary key and foreign key must be explicitly created.

role-name is the name of the foreign key. The main function of *role-name* is to distinguish two foreign keys to the same table. If no *role-name* is specified, the role name is assigned as follows:

1. If there is no foreign key with a *role-name* the same as the table name, the table name is assigned as the *role-name*.
2. If the table name is already taken, the *role-name* is the table name concatenated with a zero-padded 3-digit number unique to the table.

The referential integrity action defines the action to be taken to maintain foreign-key relationships in the database. Whenever a primary key value is changed or deleted from a database table, there may be corresponding foreign key values in other tables that should be modified in some way. You can specify an ON DELETE clause, followed by the RESTRICT clause.

- **RESTRICT** – generates an error if you try to update or delete a primary key value while there are corresponding foreign keys elsewhere in the database. Generates an error if you try to update a foreign key so that you create new values unmatched by a candidate key. This is the default action, unless you specify that LOAD optionally reject rows that violate referential integrity. This enforces referential integrity at the statement level.

If you use **CHECK ON COMMIT** without specifying any actions, then **RESTRICT** is implied as an action for **DELETE**. SAP Sybase IQ does not support **CHECK ON COMMIT**.

a global temporary table cannot have a foreign key that references a base table and a base table cannot have a foreign key that references a global temporary table. Local temporary tables cannot have or be referenced by a foreign key.

- **CHECK (condition)** – no row is allowed to fail the condition. If an **INSERT** statement would cause a row to fail the condition, the operation is not permitted and the effects of the statement are undone.

The change is rejected only if the condition is **FALSE**; in particular, the change is allowed if the condition is **UNKNOWN**. **CHECK** condition is not enforced by SAP Sybase IQ.

Note: If possible, do not define referential integrity foreign key-primary key relationships in SAP Sybase IQ unless you are certain there are no orphan foreign keys.

- **Remote Tables** – foreign-key definitions are ignored on remote tables. Foreign-key definitions on local tables that refer to remote tables are also ignored. Primary-key definitions are sent to the remote server if the server supports it.
- **PARTITION BY** – divides large tables into smaller, more manageable storage objects. Partitions share the same logical attributes of the parent table, but can be placed in separate dbspaces and managed individually. SAP Sybase IQ supports several table partitioning schemes:
 - hash-partitions
 - range-partitions
 - composite-partitions

A partition-key is the column or columns that contain the table partitioning keys. Partition keys can contain **NULL** and **DEFAULT** values, but cannot contain:

- **LOB (BLOB or CLOB)** columns
- **BINARY**, or **VARBINARY** columns
- **CHAR** or **VARCHAR** columns whose length is over 255 bytes
- **BIT** columns
- **FLOAT/DOUBLE/REAL** columns
- **PARTITION BY RANGE** – partitions rows by a range of values in the partitioning column. Range partitioning is restricted to a single partition key column and a maximum of 1024 partitions. In a range-partitioning-scheme, the partition-key is the column that contains the table partitioning keys:

```
range-partition-decl:
  partition-name VALUES <= ( {constant-expr | MAX } [ ,
```

```
{ constant-expr | MAX } ] ... )
  [ IN dbspace-name ]
```

The partition-name is the name of a new partition on which table rows are stored. Partition names must be unique within the set of partitions on a table. The partition-name is required.

- **VALUE** – specifies the inclusive upper bound for each partition (in ascending order). The user must specify the partitioning criteria for each range partition to guarantee that each row is distributed to only one partition. NULLs are allowed for the partition column and rows with NULL as partition key value belong to the first table partition. However, NULL cannot be the bound value.

There is no lower bound (MIN value) for the first partition. Rows of NULL cells in the first column of the partition key will go to the first partition. For the last partition, you can either specify an inclusive upper bound or MAX. If the upper bound value for the last partition is not MAX, loading or inserting any row with partition key value larger than the upper bound value of the last partition generates an error.

- **Max** – denotes the infinite upper bound and can only be specified for the last partition.
- **IN** – specifies the dbspace in the partition-decl on which rows of the partition should reside.

These restrictions affect partitions keys and bound values for range partitioned tables:

- Partition bounds must be constants, not constant expressions.
- Partition bounds must be in ascending order according to the order in which the partitions were created. That is, the upper bound for the second partition must be higher than for the first partition, and so on.

In addition, partition bound values must be compatible with the corresponding partition-key column data type. For example, VARCHAR is compatible with CHAR.

- If a bound value has a different data type than that of its corresponding partition key column, SAP Sybase IQ converts the bound value to the data type of the partition key column, with these exceptions:
- Explicit conversions are not allowed. This example attempts an explicit conversion from INT to VARCHAR and generates an error:

```
CREATE TABLE Employees (emp_name VARCHAR(20))
PARTITION BY RANGE (emp_name)
(p1 VALUES <= (CAST (1 AS VARCHAR(20))),
p2 VALUES <= (CAST (10 AS VARCHAR(20)))
```

- Implicit conversions that result in data loss are not allowed. In this example, the partition bounds are not compatible with the partition key type. Rounding assumptions may lead to data loss and an error is generated:

```
CREATE TABLE emp_id (id INT) PARTITION BY RANGE (id) (p1 VALUES
<= (10.5), p2 VALUES <= (100.5))
```

- In this example, the partition bounds and the partition key data type are compatible. The bound values are directly converted to float values. No rounding is required, and conversion is supported:


```
CREATE TABLE id_emp (id FLOAT)
PARTITION BY RANGE(id) (p1 VALUES <= (10),
p2 VALUES <= (100))
```

- Conversions from non-binary data types to binary data types are not allowed. For example, this conversion is not allowed and returns an error:

```
CREATE TABLE newemp (name BINARY)
PARTITION BY RANGE(name)
(p1 VALUES <= ("Maarten"),
p2 VALUES <= ("Zymmerman"))
```

- NULL cannot be used as a boundary in a range-partitioned table.
- The row will be in the first partition if the cell value of the 1st column of the partition key evaluated to be NULL. SAP Sybase IQ supports only single column partition keys, so any NULL in the partition key distributes the row to the first partition.
- **PARTITION BY HASH** – maps data to partitions based on partition-key values processed by an internal hashing function. Hash partition keys are restricted to a maximum of eight columns with a combined declared column width of 5300 bytes or less. For hash partitions, the table creator determines only the partition key columns; the number and location of the partitions are determined internally.

In a hash-partitioning declaration, the partition-key is a column or group of columns, whose composite value determines the partition where each row of data is stored:

```
hash-partitioning-scheme:
HASH ( partition-key [ , partition-key, ... ] )
```

- **Restrictions** –
 - You can only hash partition a base table. Attempting to partitioning a global temporary table or a local temporary table raises an error.
 - You cannot add, drop, merge, or split a hash partition.
 - You cannot add or drop a column from a hash partition key.
- **PARTITION BY HASH RANGE** – subpartitions a hash-partitioned table by range. In a hash-range-partitioning-scheme declaration, a **SUBPARTITION BY RANGE** clause adds a new range subpartition to an existing hash-range partitioned table:

```
hash-range-partitioning-scheme:
PARTITION BY HASH ( partition-key [ , partition-key, ... ] )
[ SUBPARTITION BY RANGE ( range-partition-decl [ , range-
partition-decl ... ] ) ]
```

The hash partition specifies how the data is logically distributed and colocated; the range subpartition specifies how the data is physically placed. The new range subpartition is logically partitioned by hash with the same hash partition keys as the existing hash-range partitioned table. The range subpartition key is restricted to one column.

- **Restrictions** –
 - You can only hash partition a base table. Attempting to partitioning a global temporary table or a local temporary table raises an error.

- You cannot add, drop, merge, or split a hash partition.
- You cannot add or drop a column from a hash partition key.

Note: Range-partitions and composite partitioning schemes, like hash-range partitions, require the separately licensed VLDB Management option.

Examples

- **Example 1** – create a table named `SalesOrders2` with five columns. Data pages for columns `FinancialCode`, `OrderDate`, and `ID` are in dbspace `Dsp3`. Data pages for integer column `CustomerID` are in dbspace `Dsp1`. Data pages for CLOB column `History` are in dbspace `Dsp2`. Data pages for the primary key, HG for `ID`, are in dbspace `Dsp4`:

```
CREATE TABLE SalesOrders2 (  
  FinancialCode CHAR(2),  
  CustomerID int IN Dsp1,  
  History CLOB IN Dsp2,  
  OrderDate TIMESTAMP,  
  ID BIGINT,  
  PRIMARY KEY(ID) IN Dsp4  
 ) IN Dsp3
```

- **Example 2** – create a table `fin_code2` with four columns. Data pages for columns `code`, `type`, and `id` are in the default dbspace, which is determined by the value of the database option `DEFAULT_DBSPACE`. Data pages for CLOB column `description` are in dbspace `Dsp2`. Data pages from foreign key `fk1`, HG for `c1` are in dbspace `Dsp4`:

```
CREATE TABLE fin_code2 (  
  code INT,  
  type CHAR(10),  
  description CLOB IN Dsp2,  
  id BIGINT,  
  FOREIGN KEY fk1(id) REFERENCES SalesOrders(ID) IN Dsp4  
 )
```

- **Example 3** – create a table `t1` where partition `p1` is adjacent to `p2` and partition `p2` is adjacent to `p3`:

```
CREATE TABLE t1 (c1 INT, c2 INT)  
PARTITION BY RANGE(c1)  
(p1 VALUES <= (0), p2 VALUES <= (10), p3 VALUES <= (100))
```

- **Example 4** – create a RANGE partitioned table `bar` with six columns and three partitions, mapping data to partitions based on dates:

```
CREATE TABLE bar (  
  c1 INT IQ UNIQUE(65500),  
  c2 VARCHAR(20),  
  c3 CLOB PARTITION (P1 IN Dsp11, P2 IN Dsp12,  
  P3 IN Dsp13),  
  c4 DATE,  
  c5 BIGINT,
```

```

c6 VARCHAR(500) PARTITION (P1 IN Dsp21,
    P2 IN Dsp22),
PRIMARY KEY (c5) IN Dsp2) IN Dsp1
PARTITION BY RANGE (c4)
(P1 VALUES <= ('2006/03/31') IN Dsp31,
 P2 VALUES <= ('2006/06/30') IN Dsp32,
 P3 VALUES <= ('2006/09/30') IN Dsp33
) ;

```

Data page allocation for each partition:

Partition	Dbspaces	Columns
P1	Dsp31	c1, c2, c4, c5
P1	Dsp11	c3
P1	Dsp21	c6
P2	Dsp32	c1, c2, c4, c5
P2	Dsp12	c3
P2	Dsp22	c6
P3	Dsp33	c1, c2, c4, c5, c6
P3	Dsp13	c3
P1, P2, P3	Dsp1	lookup store of c1 and other shared data
P1, P2, P3	Dsp2	primary key (HG for c5)

- **Example 5** – create a HASH partitioned (table tbl42) that includes a PRIMARY KEY (column c1) and a HASH PARTITION KEY (columns c4 and c3).

```

CREATE TABLE tbl42 (
    c1 BIGINT NOT NULL,
    c2 CHAR(2) IQ UNIQUE(50),
    c3 DATE IQ UNIQUE(36524),
    c4 VARCHAR(200),
    PRIMARY KEY (c1)
)
PARTITION BY HASH ( c4, c3 )

```

- **Example 6** – create a hash-ranged partitioned table with a PRIMARY KEY (column c1), a hash partition key (columns c4 and c2) and a range subpartition key (column c3).

```

CREATE TABLE tbl42 (
    c1 BIGINT NOT NULL,
    c2 CHAR(2) IQ UNIQUE(50),
    c3 DATE,
    c4 VARCHAR(200),

```

```
PRIMARY KEY (c1)) IN Dsp1

PARTITION BY HASH ( c4, c2 )
SUBPARTITION BY RANGE ( c3 )
( P1 VALUES <= (2011/03/31) IN Dsp31,
  P2 VALUES <= (2011/06/30) IN Dsp32,
  P3 VALUES <= (2011/09/30) IN Dsp33) ;
```

- **Example 7** – create a table for a library database to hold information on borrowed books:

```
CREATE TABLE borrowed_book (
date_borrowed DATE NOT NULL,
date_returned DATE,
book CHAR(20)
REFERENCES library_books (isbn),
CHECK( date_returned >= date_borrowed )
)
```

- **Example 8** – create table t1 at the remote server SERVER_A and create a proxy table named t1 that is mapped to the remote table:

```
CREATE TABLE t1
( a INT,
  b CHAR(10))
AT 'SERVER_A.db1.joe.t1'
```

- **Example 9** – create table tab1 that contains a column c1 with a default value of the special constant LAST USER:

```
CREATE TABLE tab1(c1 CHAR(20) DEFAULT LAST USER)
```

- **Example 10** – create a local temporary table tab1 that contains a column c1:

```
CREATE LOCAL TEMPORARY TABLE tab1(c1 int) IN IQ_SYSTEM_TEMP
```

The example creates tab1 in the IQ_SYSTEM_TEMP dbspace in the following cases:

- DQP_ENABLED logical server policy option is set ON but there are no read-write files in IQ_SHARED_TEMP
- DQP_ENABLED option is OFF, TEMP_DATA_IN_SHARED_TEMP logical server policy option is ON, but there are no read-write files in IQ_SHARED_TEMP
- Both the DQP_ENABLED option and the TEMP_DATA_IN_SHARED_TEMP option are set OFF

The example creates the same table tab1 in the IQ_SHARED_TEMP dbspace in the following cases:

- DQP_ENABLED is ON and there are read-write files in IQ_SHARED_TEMP
- DQP_ENABLED is OFF, TEMP_DATA_IN_SHARED_TEMP is ON, and there are read-write files in IQ_SHARED_TEMP
- **Example 11** – create a table tab1 that is enabled to use row-level versioning, and real-time storage in the in-memory RLV store.

```
CREATE TABLE tab1 ( c1 INT, c2 CHAR(25) ) ENABLE RLV STORE
```

Usage

You can create a table for another user by specifying an owner name. If **GLOBAL TEMPORARY** or **LOCAL TEMPORARY** is not specified, the table is referred to as a base table. Otherwise, the table is a temporary table.

A created global temporary table exists in the database like a base table and remains in the database until it is explicitly removed by a **DROP TABLE** statement. The rows in a temporary table are visible only to the connection that inserted the rows. Multiple connections from the same or different applications can use the same temporary table at the same time and each connection sees only its own rows. A given connection inherits the schema of a global temporary table as it exists when the connection first refers to the table. The rows of a temporary table are deleted when the connection ends.

When you create a local temporary table, omit the owner specification. If you specify an owner when creating a temporary table, for example, `CREATE TABLE dbo.#temp(coll int)`, a base table is incorrectly created.

An attempt to create a base table or a global temporary table will fail, if a local temporary table of the same name exists on that connection, as the new table cannot be uniquely identified by `owner.table`.

You can, however, create a local temporary table with the same name as an existing base table or global temporary table. References to the table name access the local temporary table, as local temporary tables are resolved first.

For example, consider this sequence:

```
CREATE TABLE t1 (c1 int);
INSERT t1 VALUES (9);

CREATE LOCAL TEMPORARY TABLE t1 (c1 int);
INSERT t1 VALUES (8);

SELECT * FROM t1;
```

The result returned is 8. Any reference to `t1` refers to the local temporary table `t1` until the local temporary table is dropped by the connection.

In a procedure, use the **CREATE LOCAL TEMPORARY TABLE** statement, instead of the **DECLARE LOCAL TEMPORARY TABLE** statement, when you want to create a table that persists after the procedure completes. Local temporary tables created using the **CREATE LOCAL TEMPORARY TABLE** statement remain until they are either explicitly dropped, or until the connection closes.

Local temporary tables created in **IF** statements using **CREATE LOCAL TEMPORARY TABLE** also persist after the **IF** statement completes.

SAP Sybase IQ does not support the **CREATE TABLE ENCRYPTED** clause for table-level encryption of SAP Sybase IQ tables. However, the **CREATE TABLE ENCRYPTED** clause is supported for SQL Anywhere tables in an SAP Sybase IQ database.

Side Effects

- Automatic commit

Standards

- SQL–Vendor extension to ISO/ANSI SQL grammar.
These are vendor extensions:
 - The { **IN** | **ON** } *dbspace-name* clause
 - The **ON COMMIT** clause
 - Some of the default values
- Supported by Adaptive Server Enterprise, with some differences.
 - Temporary tables – You can create a temporary table by preceding the table name in a **CREATE TABLE** statement with a pound sign (#). These temporary tables are SAP Sybase IQ declared temporary tables, which are available only in the current connection. For information about declared temporary tables, see *DECLARE LOCAL TEMPORARY TABLE Statement*.
 - Physical placement – Physical placement of a table is carried out differently in SAP Sybase IQ and in Adaptive Server Enterprise. The **ON segment-name** clause supported by Adaptive Server Enterprise is supported in SAP Sybase IQ, but *segment-name* refers to an IQ dbspace.
 - Constraints – SAP Sybase IQ does not support named constraints or named defaults, but does support user-defined data types that allow constraint and default definitions to be encapsulated in the data type definition. It also supports explicit defaults and **CHECK** conditions in the **CREATE TABLE** statement.
 - NULL default – By default, columns in Adaptive Server Enterprise default to **NOT NULL**, whereas in SAP Sybase IQ the default setting is **NULL**, to allow **NULL** values. This setting can be controlled using the `ALLOW_NULLS_BY_DEFAULT` option. See *ALLOW_NULLS_BY_DEFAULT Option [TSQL]*. To make your data definition statements transferable, explicitly specify **NULL** or **NOT NULL**.

Permissions

Table Type	Privileges Required
Base table in the IQ main store	<p>Table owned by self – Requires CREATE privilege on the dbspace where the table is created. Also requires one of:</p> <ul style="list-style-type: none"> • CREATE TABLE system privilege. • CREATE ANY OBJECT system privilege. <p>Table owned by any user – Requires CREATE privilege on the dbspace where the table is created. Also requires one of:</p> <ul style="list-style-type: none"> • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege.
Global temporary table	<p>Table owned by self – Requires one of:</p> <ul style="list-style-type: none"> • CREATE TABLE system privilege. • CREATE ANY OBJECT system privilege. <p>Table owned by any user – Requires one of:</p> <ul style="list-style-type: none"> • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege.
Proxy table	<p>Table owned by self – Requires one of:</p> <ul style="list-style-type: none"> • CREATE PROXY TABLE system privilege. • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege. <p>Table owned by any user – Requires one of:</p> <ul style="list-style-type: none"> • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege.

See also

- *ALLOW_NULLS_BY_DEFAULT Option [TSQL]* on page 479
- *ALTER TABLE Statement* on page 52
- *CREATE DBSPACE Statement* on page 117
- *CREATE INDEX Statement* on page 142
- *DECLARE LOCAL TEMPORARY TABLE Statement* on page 246
- *DROP Statement* on page 255
- *FP_NBIT_AUTOSIZE_LIMIT Option* on page 536
- *FP_NBIT_ENFORCE_LIMITS Option* on page 538
- *FP_NBIT_LOOKUP_MB Option* on page 540

- *FP_NBIT_ROLLOVER_MAX_MB Option* on page 542

CREATE TEXT CONFIGURATION Statement

Creates a text configuration object.

Note: This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Syntax

```
CREATE TEXT CONFIGURATION [ owner. ] new-config-name  
FROM [ owner. ] existing-config-name
```

Parameters

- **FROM** – specifies the name of a text configuration object to use as the template for creating the new text configuration object. The names of the default text configuration objects are DEFAULT_CHAR and DEFAULT_NCHAR. DEFAULT_CHAR is supported for SAP Sybase IQ tables only; DEFAULT_NCHAR is supported on SQL Anywhere tables only.

Examples

- **Example 1** – creates a text configuration object, `max_term_sixteen`, using the `default_char` text configuration object, then use **ALTER TEXT CONFIGURATION** to change the maximum term length for `max_term_sixteen` to 16:

```
CREATE TEXT CONFIGURATION max_term_sixteen FROM default_char;
```

```
ALTER TEXT CONFIGURATION max_term_sixteen MAXIMUM TERM LENGTH 16;
```

Usage

Create a text configuration object using another text configuration object as a template, then alter the options as needed using the **ALTER TEXT CONFIGURATION** statement.

To view the list of all text configuration objects and their settings in the database, query the `SYSTEXTCONFIG` system view.

Side Effects:

- Automatic commit.

Permissions

Text configuration object to be owned by self –

- Requires **CREATE TEXT CONFIGURATION** system privilege.

Text configuration object to be owned by any user – Requires one of:

- CREATE ANY TEXT CONFIGURATION system privilege.
- CREATE ANY OBJECT system privilege.

All text configuration objects have PUBLIC access. Any user with privilege to create a **TEXT** index can use any text configuration object.

CREATE TEXT INDEX Statement

Creates a TEXT index and specifies the text configuration object to use.

Note: This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Syntax

```
CREATE TEXT INDEX text-index-name
  ON [ owner.]table-name( column-name, ...)
  [ IN dbspace-name ]
  [ CONFIGURATION [ owner.]text-configuration-name ]
  [ IMMEDIATE REFRESH ]
```

Parameters

- **ON** – specifies the table and column on which to build the TEXT index.
- **IN** – specifies the dbspace in which the TEXT index is located. If this clause is not specified, then the TEXT index is created in the same dbspace as the underlying table.
- **CONFIGURATION** – specifies the text configuration object to use when creating the TEXT index. If this clause is not specified, the default_char text configuration object is used.
- **IMMEDIATE REFRESH** – (default) refreshes the TEXT index each time changes in the underlying table impact data in the TEXT index. Only permitted value for tables in SAP Sybase IQ main store. Once created, the IMMEDIATE REFRESH clause cannot be changed.

Examples

- **Example 1** – creates a TEXT index, myTxtIdx, on the CompanyName column of the Customers table in the iqdemo database, using the max_term_sixteen text configuration object:

```
CREATE TEXT INDEX myTxtIdx ON Customers (CompanyName );
CONFIGURATION max_term_sixteen;
```

Usage

You cannot create a TEXT index on views or temporary tables, or on an IN SYSTEM materialized view. The **BEGIN PARALLEL IQ...END PARALLEL IQ** statement does not support **CREATE TEXT INDEX**.

Side Effects:

- Automatic commit.

Permissions

Requires one of:

- CREATE ANY INDEX system privilege along with CREATE privilege on the dbspace where the index is being created.
- CREATE ANY OBJECT system privilege.

CREATE TRIGGER statement

Creates a trigger on a table.

Syntax

```
CREATE [ OR REPLACE ] TRIGGER trigger-name trigger-type
{ trigger-event-list | UPDATE OF column-list }
[ ORDER integer ] ON table-name
[ REFERENCING [ OLD AS old-name ]
  [ NEW AS new-name ]
  [ REMOTE AS remote-name ] ]
[ FOR EACH { ROW | STATEMENT } ]
[ WHEN ( search-condition ) ]
trigger-body
```

```
column-list : column-name[, ...]
```

```
trigger-type :
BEFORE
| AFTER
| INSTEAD OF
| RESOLVE
```

```
trigger-event-list : trigger-event[, ... ]
```

```
trigger-event :
DELETE
| INSERT
| UPDATE ( column-name )
| UPDATING [ ( column-name-string ) ]
```

```
trigger-body : a BEGIN statement.
```

Parameters

OR REPLACE clause – Specifying OR REPLACE creates a new trigger, or replaces an existing trigger with the same name.

trigger-event – Triggers can be fired by the following events. You can define either multiple triggers for DELETE, INSERT, or UPDATE events, or one trigger for an UPDATE OF *column-list* event:

- **DELETE clause** – Invoked whenever a row of the associated table is deleted.
- **INSERT clause** – Invoked whenever a new row is inserted into the table associated with the trigger.
- **UPDATE clause** – Invoked whenever a row of the associated table is updated.

If you specify an UPDATE clause, you must also supply a REFERENCING clause to avoid syntax errors.

- **UPDATE OF *column-list* clause** – Invoked whenever a row of the associated table is updated and a column in the *column-list* is modified. This type of trigger event cannot be used in a *trigger-event-list*; it must be the only trigger event defined for the trigger. This clause cannot be used in an INSTEAD OF trigger.

You can write separate triggers for each event that you need to handle or, if you have some shared actions and some actions that depend on the event, you can create a trigger for all events and use an IF statement to distinguish the action taking place.

- **UPDATING clause** – The argument for UPDATING is a quoted string (for example, UPDATING ('mycolumn')). The argument for UPDATE is an identifier (for example, UPDATE (mycolumn)). The two versions are interoperable, and are included for compatibility with SQL dialects of other vendors' DBMS.

If you specify an UPDATING clause, then you must also supply a REFERENCING clause to avoid syntax errors.

trigger-type – Row-level triggers can be defined to execute BEFORE, AFTER, or INSTEAD OF an insert, update, or delete operation. Statement-level triggers can be defined to execute INSTEAD OF or AFTER the statement.

BEFORE UPDATE triggers fire any time an UPDATE occurs on a row, whether the new value differs from the old value. That is, if a *column-list* is specified for a BEFORE UPDATE trigger, then the trigger fires if any of the columns in *column-list* appear in the SET clause of the UPDATE statement. If a *column-list* is specified for an AFTER UPDATE trigger, then the trigger is fired only if the value of any of the columns in *column-list* is *changed* by the UPDATE statement.

INSTEAD OF triggers are the only form of trigger that you can define on a regular view. INSTEAD OF triggers replace the triggering action with another action. When an INSTEAD OF trigger fires, the triggering action is skipped and the specified action is performed. INSTEAD OF triggers can be defined as a row-level or a statement-level trigger. A statement-level INSTEAD OF trigger replaces the entire statement, including all row-level operations. If

a statement-level **INSTEAD OF** trigger fires, then no row-level triggers fire as a result of that statement. However, the body of the statement-level trigger could perform other operations that, in turn, cause other row-level triggers to fire.

If you are defining an **INSTEAD OF** trigger, then you cannot use the **UPDATE OF *column-list*** clause, the **ORDER** clause, or the **WHEN** clause.

ORDER clause – When defining additional triggers of the same type (insert, update, or delete) to fire at the same time (before, after, or resolve), you must specify an **ORDER** clause to tell the database server the order in which to fire the triggers. Order numbers must be unique among same-type triggers configured to fire at the same time. If you specify an order number that is not unique, then an error is returned. Order numbers do not need to be in consecutive order (for example, you could specify 1, 12, 30). The database server fires the triggers starting with the lowest number.

If you omit the **ORDER** clause, or specify 0, then the database server assigns the order of 1. However, if another same-type trigger is already set to 1, then an error is returned.

When adding additional triggers, you may need to modify the existing same-type triggers for the event, depending on whether the actions of the triggers interact. If they do not interact, then the new trigger must have an **ORDER** value higher than the existing triggers. If they do interact, you need to consider what the other triggers do, and you may need to change the order in which they fire.

The **ORDER** clause is not supported for **INSTEAD OF** triggers since there can only be one **INSTEAD OF** trigger of each type (insert, update, or delete) defined on a table or view.

REFERENCING clause – The **REFERENCING OLD** and **REFERENCING NEW** clauses allow you to refer to the inserted, deleted, or updated rows. With this clause an **UPDATE** is treated as a delete followed by an insert.

An **INSERT** takes the **REFERENCING NEW** clause, which represents the inserted row. There is no **REFERENCING OLD** clause.

A **DELETE** takes the **REFERENCING OLD** clause, which represents the deleted row. There is no **REFERENCING NEW** clause.

An **UPDATE** takes the **REFERENCING OLD** clause, which represents the row before the update, and it takes the **REFERENCING NEW** clause, which represents the row after the update.

The meanings of **REFERENCING OLD** and **REFERENCING NEW** differ, depending on whether the trigger is a row-level or a statement-level trigger. For row-level triggers, the **REFERENCING OLD** clause allows you to refer to the values in a row before an update or delete, and the **REFERENCING NEW** clause allows you to refer to the inserted or updated values. The **OLD** and **NEW** rows can be referenced in **BEFORE** and **AFTER** triggers. The **REFERENCING NEW** clause allows you to modify the new row in a **BEFORE** trigger before the insert or update operation takes place.

For statement-level triggers, the REFERENCING OLD and REFERENCING NEW clauses refer to declared temporary tables holding the old and new values of the rows.

FOR EACH clause – To declare a trigger as a row-level trigger, use the FOR EACH ROW clause. To declare a trigger as a statement-level trigger, you can either use a FOR EACH STATEMENT clause or omit the FOR EACH clause. For clarity, it is recommended that you specify the FOR EACH STATEMENT clause if you are declaring a statement-level trigger.

WHEN clause – The trigger fires only for rows where the search-condition evaluates to true. The WHEN clause can be used only with row level triggers. This clause cannot be used in an INSTEAD OF trigger.

trigger-body – The trigger body contains the actions to take when the triggering action occurs, and consists of a BEGIN statement.

You can include trigger operation conditions in the BEGIN statement. Trigger operation conditions perform actions depending on the trigger event that caused the trigger to fire. For example, if the trigger is defined to fire for both updates and deletes, you can specify different actions for the two conditions.

Remarks

The CREATE TRIGGER statement creates a trigger associated with a table in the database, and stores the trigger in the database.

You cannot define a trigger on a materialized view. If you do, a SQLE_INVALID_TRIGGER_MATVIEW error is returned.

A trigger is declared as either a row-level trigger, in which case it executes before or after each row is modified, or a statement-level trigger, in which case it executes after the entire triggering statement is completed.

CREATE TRIGGER puts a table lock on the table and requires exclusive use of the table.

Privileges

You must have the CREATE ANY TRIGGER or CREATE ANY OBJECT system privilege. Additionally, you must be the owner of the table the trigger is built on or have one of the following privileges:

- ALTER privilege on the table
- ALTER ANY TABLE system privilege
- ALTER ANY OBJECT system privilege

To create a trigger on a view owned by someone else, you must have either the CREATE ANY TRIGGER or CREATE ANY OBJECT system privilege, and you must have either the ALTER ANY VIEW or ALTER ANY OBJECT system privilege.

Side effects

Automatic commit.

Standards and compatibility

- **SQL/2008** – CREATE TRIGGER is part of optional SQL language feature T211 "Basic trigger capability" of the SQL/2008 standard. Row triggers are optional SQL language feature T212, while INSTEAD OF triggers are optional SQL language feature T213.

Some features of SAP Sybase IQ triggers are vendor extensions. These include:

- The optional OR REPLACE syntax. If an existing trigger is replaced, authorization of the creation of the new trigger instance is bypassed.
- The ORDER clause. In SQL/2008, triggers are fired in the order they were created.
- RESOLVE triggers are a vendor extension.
- **Transact-SQL** – ROW and RESOLVE triggers are not supported by Adaptive Server Enterprise. The SAP Sybase IQ Transact-SQL dialect does not support Transact-SQL INSTEAD OF triggers, though these are supported by Adaptive Server Enterprise. Transact-SQL triggers are defined using different syntax.

Example

This example creates a statement-level trigger. First, create a table as shown in this CREATE TABLE statement (requires the CREATE TABLE system privilege):

```
CREATE TABLE t0
( id INTEGER NOT NULL,
  times TIMESTAMP NULL DEFAULT CURRENT_TIMESTAMP,
  remarks TEXT NULL,
  PRIMARY KEY ( id )
);
```

Next, create a statement-level trigger for this table:

```
CREATE TRIGGER myTrig AFTER INSERT ORDER 4 ON t0
REFERENCING NEW AS new_name
FOR EACH STATEMENT
BEGIN
  DECLARE @id1 INTEGER;
  DECLARE @times1 TIMESTAMP;
  DECLARE @remarks1 LONG VARCHAR;
  DECLARE @err_notfound EXCEPTION FOR SQLSTATE VALUE '02000';
  //declare a cursor for table new_name
  DECLARE new1 CURSOR FOR
    SELECT id, times, remarks FROM new_name;
  OPEN new1;
  //Open the cursor, and get the value
  LoopGetRow:
  LOOP
    FETCH NEXT new1 INTO @id1, @times1, @remarks1;
    IF SQLSTATE = @err_notfound THEN
      LEAVE LoopGetRow
    END IF;
    //print the value or for other use
    PRINT (@remarks1);
  END LOOP LoopGetRow;
```

```
CLOSE new1
END;
```

The following example replaces the myTrig trigger created in the previous example.

```
CREATE OR REPLACE TRIGGER myTrig AFTER INSERT ORDER 4 ON t0
REFERENCING NEW AS new_name
FOR EACH STATEMENT
BEGIN
FOR l1 AS new1 CURSOR FOR
    SELECT id, times, remarks FROM new_name
DO
    //print the value or for other use
    PRINT (@remarks1);
END FOR;
END;
```

The next example shows how you can use REFERENCING NEW in a BEFORE UPDATE trigger. This example ensures that postal codes in the new Employees table are in uppercase. You must have SELECT, ALTER, and UPDATE privileges on GROUPO.Employees to execute this statement:

```
CREATE TRIGGER emp_upper_postal_code
BEFORE UPDATE OF PostalCode
ON GROUPO.Employees
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN ( ISNUMERIC( new_emp.PostalCode ) = 0 )
BEGIN
    -- Ensure postal code is uppercase (employee might be
    -- in Canada where postal codes contain letters)
    SET new_emp.PostalCode = UPPER(new_emp.PostalCode)
END;

UPDATE GROUPO.Employees SET state='ON', PostalCode='n2x 4y7' WHERE
EmployeeID=191;
SELECT PostalCode FROM GROUPO.Employees WHERE EmployeeID = 191;
```

The next example shows how you can use REFERENCING OLD in a BEFORE DELETE trigger. This example prevents deleting an employee from the Employees table who has not been terminated.

```
CREATE TRIGGER TR_check_delete_employee
BEFORE DELETE
ON Employees
REFERENCING OLD AS current_employee
FOR EACH ROW WHEN ( current_employee.Terminate IS NULL )
BEGIN
    RAISERROR 30001 'You cannot delete an employee who has not been
fired';
END;
```

The next example shows how you can use REFERENCING NEW and REFERENCING OLD in a BEFORE UPDATE trigger. This example prevents a decrease in an employee's salary.

SQL Statements

```
CREATE TRIGGER TR_check_salary_decrease
  BEFORE UPDATE
    ON GROUPO.Employees
  REFERENCING OLD AS before_update
  NEW AS after_update
FOR EACH ROW
BEGIN
  IF after_update.salary < before_update.salary THEN
    RAISERROR 30002 'You cannot decrease a salary';
  END IF;
END;
```

The next example shows how you can use **REFERENCING NEW** in a **BEFORE INSERT** and **UPDATE** trigger. The following example creates a trigger that fires before a row in the **SalesOrderItems** table is inserted or updated.

```
CREATE TRIGGER TR_update_date
  BEFORE INSERT, UPDATE
    ON GROUPO.SalesOrderItems
  REFERENCING NEW AS new_row
FOR EACH ROW
BEGIN
  SET new_row.ShipDate = CURRENT_TIMESTAMP;
END;
```

The following trigger displays a message on the **Messages** tab of the **Interactive SQL Results** pane showing which action caused the trigger to fire.

```
CREATE TRIGGER tr BEFORE INSERT, UPDATE, DELETE
ON sample_table
REFERENCING OLD AS t1old
FOR EACH ROW
BEGIN
  DECLARE msg varchar(255);
  SET msg = 'This trigger was fired by an ';
  IF INSERTING THEN
    SET msg = msg || 'insert'
  ELSEIF DELETING THEN
    set msg = msg || 'delete'
  ELSEIF UPDATING THEN
    set msg = msg || 'update'
  END IF;
  MESSAGE msg TO CLIENT
END;
```

CREATE USER Statement

Creates a user.

Syntax

```
CREATE USER user-name [ IDENTIFIED BY password ]
  [ LOGIN POLICY policy-name ]
  [ FORCE PASSWORD CHANGE { ON | OFF } ]
```


Parameters

- **user-name** – name of the user.
- **IDENTIFIED BY** – the password for the user.
- **policy-name** – name of the login policy to assign the user. No change is made if you do not specify a login policy.
- **FORCE PASSWORD CHANGE** – controls whether the user must specify a new password upon logging in. This setting overrides the `PASSWORD_EXPIRY_ON_NEXT_LOGIN` option setting in the user's login policy.

Note: This functionality is not currently implemented when logging in to Sybase Control Center. A user will not be prompted to change their password. He or she will be prompted, however, when logging in to SAP Sybase IQ outside of Sybase Control Center (for example, using Interactive SQL).

- **password** – You do not have to specify a password for the user. A user without a password cannot connect to the database. This is useful if you are creating a role and do not want anyone to connect to the database using the role user ID. A user ID must be a valid identifier. User IDs and passwords cannot:
 - Begin with white space, single quotes, or double quotes
 - End with white space
 - Contain semicolons

A password can be either a valid identifier, or a string (maximum 255 characters) placed in single quotes. Passwords are case-sensitive. The password should be composed of 7-bit ASCII characters, as other characters may not work correctly if the database server cannot convert them from the client's character set to UTF-8.

You can use the `VERIFY_PASSWORD_FUNCTION` option to specify a function to implement password rules (for example, passwords must include at least one digit). If you do use a password verification function, you cannot specify more than one user ID and password in the **GRANT CONNECT** statement.

The encryption algorithm used for hashing the user passwords is FIPS-certified encryption support:

- The DLL is called `dbfips10.dll`
- The `HASH` function accepts the algorithms: `SHA1_FIPS` `SHA256_FIPS`
- If the `-fips` server option is specified and an algorithm that is not FIPS-certified is given to the `HASH` function, the database server uses `SHA1_FIPS` instead of `SHA1`, `SHA256_FIPS` instead of `SHA256`, and returns an error if `MD5` is used (`MD5` is not a FIPS-certified algorithm).
- If the `-fips` option is specified, the database server uses `SHA256_FIPS` for password hashing.

Examples

- **Example 1** – creates a user named `SQLTester` with the password `welcome`. The `SQLTester` user is assigned to the `Test1` login policy and the password expires on the next login:

```
CREATE USER SQLTester IDENTIFIED BY welcome
LOGIN POLICY Test1
FORCE PASSWORD CHANGE ON;
```

Standards

- SQL – Vendor extension to ISO/ANSI SQL grammar.
- Sybase – Not supported by Adaptive Server Enterprise.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *COMMENT Statement* on page 96
- *CREATE LOGIN POLICY Statement* on page 155
- *DROP LOGIN POLICY Statement* on page 263
- *DROP USER Statement* on page 275
- *GRANT ROLE Statement* on page 312
- *GRANT System Privilege Statement* on page 319
- *VERIFY_PASSWORD_FUNCTION Option* on page 659
- *ALTER LOGIN POLICY Statement* on page 24

CREATE VARIABLE Statement

Creates a new variable of the specified data type.

Syntax

```
CREATE [OR REPLACE] VARIABLE identifier data-type
    [{= | DEFAULT} initial-value]
```

```
initial-value:
    special-value
    | string
    | [ - ] number
    | ( constant-expression )
    | built-in-function ( constant-expression )
    | NULL
```

```
special-value:
```

```

CURRENT
{
  DATABASE
  | DATE
  | PUBLISHER
  | TIME
  | TIMESTAMP
  | USER
  | UTC TIMESTAMP }
| USER

```

Parameters

- **OR REPLACE** – specifying the OR REPLACE clause drops the named variable if it already exists and replaces its definition. You can use the OR REPLACE clause as an alternative to the **VAREXISTS** function in SQL scripts.
- **initial-value** – the data type must match the type defined by *data-type*.

If you specify *initial-value*, the variable is set to that value. If you do not specify *initial-value*, the variable contains the NULL value until a **SET** statement assigns a different value.

Examples

- **Example 1** – this code fragment inserts a large text value into the database:

```

EXEC SQL BEGIN DECLARE SECTION;
char buffer[5000];
EXEC SQL END DECLARE SECTION;
EXEC SQL CREATE VARIABLE hold_blob VARCHAR;
EXEC SQL SET hold_blob = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( buffer, 1, 5000, fp );
    if( size <= 0 ) break;
    /* add data to blob using concatenation
    Note that concatenation works for binary
    data too! */
    EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_blob );
EXEC SQL DROP VARIABLE hold_blob;

```

Usage

A variable can be used in a SQL expression anywhere a column name is allowed. If a column name exists with the same name as the variable, the variable value is used.

A variable can be used in a SQL expression anywhere a column name is allowed. Name resolution is performed as follows:

- Match any aliases specified in the query's SELECT list.

SQL Statements

- Match column names for any referenced tables.
- Assume the name is a variable.

Variables belong to the current connection, and disappear when you disconnect from the database, or when you use the **DROP VARIABLE** statement. Variables are not visible to other connections. **COMMIT** or **ROLLBACK** statements do not affect variables.

Variables created with the **CREATE VARIABLE** statement persist for a connection even when the statement is issued within a (**BEGIN...END**) statement. You must use **DECLARE** to create variables that only persist within a (**BEGIN...END**) statement, for example, within stored procedures.

Variables are useful for creating large text or binary objects for **INSERT** or **UPDATE** statements from Embedded SQL programs.

Local variables in procedures and triggers are declared within a compound statement.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84
- *DECLARE Statement* on page 238
- *DROP VARIABLE Statement* on page 276
- *SET Statement [ESQL]* on page 426

CREATE VIEW Statement

Creates a view on the database. Views are used to give a different perspective on the data even though it is not stored that way.

Syntax

```
CREATE [ OR REPLACE ] VIEW
... [ owner.]view-name [ ( column-name [ , ... ] ) ]
... AS select-without-order-by
... [ WITH CHECK OPTION ]
```

Parameters

- **OR REPLACE** – replaces an existing view with the same name. Existing permissions are preserved, but **INSTEAD OF** triggers on the view are dropped.
- **column-name** – specifies the names of the columns in the view. If the column name list is not specified, the view columns are given names from the select list items. To use the names from the select list items, each item must be a simple column name or have an alias name specified (see *SELECT Statement*).
- **view-name** – The default owner of a view is the current user ID. A view name can be used in place of a table name in **SELECT**, **DELETE**, **UPDATE**, and **INSERT** statements. Views, however, do not physically exist in the database as tables. They are derived each time they are used. The view is derived as the result of the **SELECT** statement specified in the **CREATE VIEW** statement. Table names used in a view should be qualified by the user ID of the table owner. Otherwise, a different user ID might not be able to find the table or might get the wrong table.
- **AS** – the **SELECT** statement on which the view is based must not contain an **ORDER BY** clause, a subquery in the **SELECT** list, or a **TOP** or **FIRST** qualification. It may have a **GROUP BY** clause and may be a **UNION**.
- **WITH CHECK OPTION** – rejects any updates and inserts to the view that do not meet the criteria of the views as defined by its **SELECT** statement. However, SAP Sybase IQ currently ignores this option (it supports the syntax for compatibility reasons).

Examples

- **Example 1** – creates a view showing all information for male employees only. This view has the same column names as the base table:

```
CREATE VIEW male_employee
AS SELECT *
FROM Employees
WHERE Sex = 'M'
```

- **Example 2** – creates a view showing employees and the departments to which they belong:

```
CREATE VIEW emp_dept
AS SELECT Surname, GivenName, DepartmentName
FROM Employees JOIN Departments
ON Employees.DepartmentID = Departments.DepartmentID
```

Usage

You cannot add or drop **IDENTIFY** or **AUTOINCREMENT** columns from a view.

Views can be updated unless the **SELECT** statement defining the view contains a **GROUP BY** clause, an aggregate function, or involves a **UNION** operation. An update to the view causes the underlying tables to be updated.

Side Effects

SQL Statements

- Automatic commit

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

View to be owned by self – Requires CREATE VIEW system privilege. Also requires one of:

- SELECT ANY TABLE system privilege.
- SELECT object permission on the underlying tables of the view.

View to be owned by any user – Requires one of:

- CREATE ANY VIEW system privilege.
- CREATE ANY OBJECT system privilege.
- Also requires one of:
 - SELECT ANY TABLE system privilege.
 - SELECT object privilege on the underlying tables of the view.

Materialized view to be owned by self – Requires CREATE MATERIALIZED VIEW system privilege. Also requires one of:

- CREATE ANY OBJECT system privilege.
- CREATE privilege on the dbspace where the materialized view is being created.
- Also requires one of:
 - SELECT ANY TABLE system privilege.
 - SELECT privilege on the underlying tables of the materialized view.

Materialized view to be owned by any user – Requires one of:

- CREATE ANY MATERIALIZED VIEW system privilege.
- CREATE ANY OBJECT system privilege.
- Also requires one of:
 - CREATE ANY OBJECT system privilege.
 - CREATE privilege on the dbspace where the materialized view is being created.

And also requires one of:

- SELECT ANY TABLE system privilege.
- SELECT privilege on the underlying tables of the materialized view.

See also

- *CREATE TABLE Statement* on page 205
- *DROP Statement* on page 255

- *SELECT Statement* on page 417

DEALLOCATE DESCRIPTOR Statement [ESQL]

Frees memory associated with a SQL descriptor area.

Syntax

```
DEALLOCATE DESCRIPTOR descriptor-name:
    string
```

Examples

- **Example 1** – see *ALLOCATE DESCRIPTOR Statement [ESQL]*.

Usage

Frees all memory associated with a descriptor area, including the data items, indicator variables, and the structure itself.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *ALLOCATE DESCRIPTOR Statement [ESQL]* on page 5
- *SET DESCRIPTOR Statement [ESQL]* on page 430

Declaration Section [ESQL]

Declares host variables in an Embedded SQL program. Host variables are used to exchange data with the database.

Syntax

```
EXEC SQL BEGIN DECLARE SECTION;
... C declarations
EXEC SQL END DECLARE SECTION;
```

Examples

- **Example 1 –**

```
EXEC SQL BEGIN DECLARE SECTION;
char *emp_lname, initials[5];
int dept;
EXEC SQL END DECLARE SECTION;
```

Usage

A declaration section is a section of C variable declarations surrounded by the **BEGIN DECLARE SECTION** and **END DECLARE SECTION** statements. A declaration section makes the SQL preprocessor aware of C variables that are used as host variables. Not all C declarations are valid inside a declaration section.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84

DECLARE Statement

Declares a SQL variable within a compound statement (**BEGIN... END**).

Syntax

DECLARE

```
variable_name [ , ... ]
data-type [(
=
| DEFAULT}
initial-value]
```

initial-value:

```
special-value
| string
| [ - ] number
| ( constant-expression )
| built-in-function ( constant-expression )
| NULL
```

special-value:

```
CURRENT
```



```
{ DATABASE
  | DATE
  | PUBLISHER
  | TIME
  | TIMESTAMP
  | USER
  | UTC TIMESTAMP }
| USER
```

Parameters

- **initial-value** – the variable is set to that value. The data type must match the type defined by *data-type*. If you do not specify an **initial-value**, the variable contains the NULL value until a **SET** statement assigns a different value.

Examples

- **Example 1** – this batch illustrates the use of the **DECLARE** statement and prints a message on the server window:

```
BEGIN
  DECLARE varname CHAR(61);
  SET varname = 'Test name';
  MESSAGE varname;
END
```

Usage

Use the **DECLARE** statement to declare variables used in the body of a procedure. The variable persists for the duration of the compound statement in which it is declared and must be unique within the compound statement.

The body of a procedure is a compound statement, and variables must be declared immediately following **BEGIN**. In a Transact-SQL procedure or trigger, there is no such restriction.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Adaptive Server Enterprise.
 - To be compatible with Adaptive Server Enterprise, the variable name must be preceded by an @.
 - In Adaptive Server Enterprise, a variable that is declared in a procedure or trigger exists for the duration of the procedure or trigger. In SAP Sybase IQ, if a variable is declared inside a compound statement, it exists only for the duration of that compound statement (whether it is declared in an SAP Sybase IQ SQL or Transact-SQL compound statement).

Permissions

None

See also

- *BEGIN ... END Statement* on page 84

DECLARE CURSOR Statement [ESQL] [SP]

Declares a cursor. Cursors are the primary means for manipulating the results of queries.

Syntax

```
DECLARE cursor-name
  [ SCROLL
    | NO SCROLL
    | DYNAMIC SCROLL
  ]
  CURSOR FOR
  { select-statement FOR for-clause
    | statement-name
    | USING variable-name }
```

```
for-clause:
  READ ONLY | UPDATE
```

Parameters

- **statement-name** – identifier or host-variable. Statements are named using the **PREPARE** statement. Cursors can be declared only for a prepared **SELECT** or **CALL**.
- **SCROLL** – a cursor declared as **SCROLL** supports the **NEXT**, **PRIOR**, **FIRST**, **LAST**, **ABSOLUTE**, and **RELATIVE** options of the **FETCH** statement. A **SCROLL** cursor lets you fetch an arbitrary row in the result set while the cursor is open.
- **NO SCROLL** – a cursor declared as **NO SCROLL** is restricted to moving forward through the result set using only the **FETCH NEXT** and **FETCH ABSOLUTE (0)** seek operations.
- **DYNAMIC SCROLL** – a cursor declared as **DYNAMIC SCROLL** supports the **NEXT**, **PRIOR**, **FIRST**, **LAST**, **ABSOLUTE**, and **RELATIVE** clauses of the **FETCH** statement. A **DYNAMIC SCROLL** cursor lets you fetch an arbitrary row in the result set while the cursor is open.

Since rows cannot be returned to once the cursor leaves the row, there are no sensitivity restrictions on the cursor. Consequently, when a **NO SCROLL** cursor is requested, SAP Sybase IQ supplies the most efficient kind of cursor, which is an asensitive cursor.

- **READ ONLY** – (default) A cursor declared **FOR READ ONLY** may not be used in a positioned **UPDATE** or a positioned **DELETE** operation.

A cursor declared **FOR READ ONLY** sees the version of table(s) on which the cursor is declared when the cursor is opened, not the version of table(s) at the time of the first **FETCH**.

For example, when the cursor is fetched, only one row can be fetched from the table:

```
CREATE TABLE t1 ( c1 INT );
INSERT t1 VALUES ( 1 );

BEGIN
DECLARE t1_cursor CURSOR FOR SELECT * FROM t1
FOR READ ONLY;
OPEN t1_cursor;
INSERT t1 VALUES ( 2 );
FETCH T1_CURSOR;
END
```

- **UPDATE** – you can update the cursor result set of a cursor declared **FOR UPDATE**. Only asensitive behavior is supported for updatable cursors; any other sensitivity is ignored.

When the cursor is opened, exclusive table locks are taken on all tables that are opened for update. Standalone **LOAD TABLE**, **UPDATE**, **INSERT**, **DELETE**, and **TRUNCATE** statements are not allowed on tables that are opened for update in the same transaction, since SAP Sybase IQ permits only one statement to modify a table at a time. You can open only one updatable cursor on a specific table at a time.

Updatable cursors are allowed to scroll, except over Open Client.

- **USING** – you can declare a cursor on a variable in stored procedures and user-defined functions. The variable is a string containing a **SELECT** statement for the cursor. The variable must be available when the **DECLARE** is processed, and so must be one of the following.

A parameter to the procedure. For example:

```
create function get_row_count(in qry varchar)
returns int
begin
    declare crsr cursor using qry;
    declare rowcnt int;

    set rowcnt = 0;
    open crsr;
    lp: loop
        fetch crsr;
        if SQLCODE <> 0 then leave lp end if;
        set rowcnt = rowcnt + 1;
    end loop;
    return rowcnt;
end
```

Nested inside another **BEGIN...END** after the variable has been assigned a value. For example:

SQL Statements

```
create procedure get_table_name(
  in id_value int, out tabname char(128))

begin
  declare qry varchar;

  set qry = 'select table_name from SYS.ISYSTAB ' ||
            'where table_id=' || string(id_value);
  begin
    declare crsr cursor using qry;

    open crsr;
    fetch crsr into tabname;
    close crsr;
  end
end
```

Examples

- **Example 1** – declares a scroll cursor in Embedded SQL:

```
EXEC SQL DECLARE cur_employee SCROLL CURSOR
FOR SELECT * FROM Employees;
```

- **Example 2** – declares a cursor for a prepared statement in Embedded SQL:

```
EXEC SQL PREPARE employee_statement
FROM 'SELECT emp_lname FROM Employees';
EXEC SQL DECLARE cur_employee CURSOR
FOR employee_statement ;
```

- **Example 3** – uses cursors in a stored procedure:

```
BEGIN
  DECLARE cur_employee CURSOR FOR
    SELECT emp_lname
    FROM Employees;
  DECLARE name CHAR(40);
  OPEN cur_employee;
  LOOP
    FETCH NEXT cur_employee INTO name;
    ...
  END LOOP;
  CLOSE cur_employee;
END
```

Usage

The **DECLARE CURSOR** statement declares a cursor with the specified name for a **SELECT** statement or a **CALL** statement.

Embedded SQL

Statements are named using the **PREPARE** statement. Cursors can be declared only for a prepared **SELECT** or **CALL**.

Updatable Cursor Support

SAP Sybase IQ supports updatable cursors.

SAP Sybase IQ supports one type of cursor sensitivity, which is defined in terms of which changes to underlying data are visible. All SAP Sybase IQ cursors are asensitive, which means that changes might be reflected in the membership, order, or values of the result set seen through the cursor, or might not be reflected at all.

With an asensitive cursor, changes effected by positioned **UPDATE** and positioned **DELETE** statements are visible in the cursor result set, except where client-side caching prevents seeing these changes. Inserted rows are not visible.

Rows that are updated so that they no longer meet the requirements of the **WHERE** clause of the open cursor are still visible.

When using cursors, there is always a trade-off between efficiency and consistency. Asensitive cursors provide efficient performance at the expense of consistency.

SAP Sybase IQ supports updatable cursors on single tables.

`LONG VARCHAR` and `LONG BINARY` data types are not supported in updatable cursors.

Scalar user-defined functions and user-defined aggregate functions are not supported in updatable cursors.

Supported query specifications for updatable cursors in SAP Sybase IQ are:

- Expressions in the select list against columns that are not functionally dependent on columns being updated
- Arbitrary subqueries with asensitive behavior, that is, changes to data referenced by subqueries are not visible in the cursor result set
- **ORDER BY** clause; the **ORDER BY** columns may be updated, but the result set does not reorder
- Columns that meet these requirements:
 - No **CAST** on a column
 - Base columns of a base table in the **SELECT** clause
 - There are no expressions or functions on that column in the **SELECT** clause and it is not duplicated in the select list (for example, `SELECT c1, c1`).
 - Base columns of a base table restricted to those listed in the **FOR UPDATE OF *column-name-list*** clause, if the clause is specified.

SAP Sybase IQ does not permit updatable cursors on queries that contain any operator that precludes a one-to-one mapping of result set rows to rows in a base table; specifically:

- **SELECT DISTINCT**
- Operator that has a **UNION**
- Operator that has a **GROUP BY**

SQL Statements

- Operator that has a **SET** function
- Operator that has an **OLAP** function, with the exception of **RANK()**

See the description of the *UPDATE (positioned) Statement [ESQL] [SP]* for information on the columns and expressions allowed in the **SET** clause for the update of a row in the result set of a cursor.

SAP Sybase IQ supports inserts only on updatable cursors where all nonnullable, nonidentity columns are both selected and updatable.

In SAP Sybase IQ, **COMMIT** and **ROLLBACK** are not allowed inside an open updatable cursor, even if the cursor is opened as a hold cursor. SAP Sybase IQ does support **ROLLBACK TO SAVEPOINT** inside an updatable cursor.

Any failure that occurs after the cursor is open results in a rollback of all operations that have been performed through this open cursor.

Updatable Cursor Limitations

A declared cursor is read-only and not updatable in cases where:

- The data extraction facility is enabled with the `TEMP_EXTRACT_NAME1` option set to a pathname
- `ANSI_CLOSE_CURSORS_ON_ROLLBACK` is set **OFF**
- `CHAINED` is set **OFF**
- The statement is **INSERT SELECT** or **SELECT INTO**
- More than one table is included
- No updatable columns exist

If SAP Sybase IQ fails to set an updatable cursor when requested, see the `.iqmsg` file for related information.

There is a limitation regarding updatable cursors and ODBC. A maximum of 65535 rows or records can be updated, deleted, or inserted at a time using these ODBC functions:

- **SQLSetPos** `SQL_UPDATE`, `SQL_DELETE`, and `SQL_ADD`
- **SQLBulkOperations** `SQL_ADD`, `SQL_UPDATE_BY_BOOKMARK`, and `SQL_DELETE_BY_BOOKMARK`

There is an implementation-specific limitation to the maximum value in the statement attribute that controls the number of effected rows to the largest value of an `UNSIGNED SMALL INT`, which is 65535.

```
SQLSetStmtAttr(HANDLE, SQL_ATTR_ROW_ARRAY_SIZE, VALUE, 0)
```

Updatable Cursor Differences

SAP Sybase IQ updatable cursors differ from ANSI SQL3 standard behavior as follows:

- Hold cursor update close on commit.

- SAP Sybase IQ locks tables when the cursor is open.
- All updates, deletes, and insert operations are applied when the cursor is closed, in this order: deletes first, then updates, then inserts.

Note: Use the **sp_iqcursorinfo** system procedure to display detailed information about cursors currently open on the server.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *CALL Statement* on page 90
- *DELETE (positioned) Statement [ESQL] [SP]* on page 250
- *FETCH Statement [ESQL] [SP]* on page 283
- *OPEN Statement [ESQL] [SP]* on page 369
- *PREPARE Statement [ESQL]* on page 375
- *SELECT Statement* on page 417
- *UPDATE (positioned) Statement [ESQL] [SP]* on page 451

DECLARE CURSOR Statement [T-SQL]

Declares a cursor that is compatible with Adaptive Server Enterprise.

Syntax

```
DECLARE cursor-name
... CURSOR FOR select-statement
...[ FOR { READ ONLY | UPDATE } ]
```

Usage

SAP Sybase IQ supports a **DECLARE CURSOR** syntax that is not supported in Adaptive Server Enterprise. For information on the full **DECLARE CURSOR** syntax, see *DECLARE CURSOR Statement [ESQL] [SP]*.

Note: Use the **sp_iqcursorinfo** system procedure to display detailed information about cursors currently open on the server.

Standards

- SQL—The FOR UPDATE and FOR READ ONLY options are Transact-SQL extensions to ISO/ANSI SQL grammar.
- Sybase—There are some features of the Adaptive Server Enterprise **DECLARE CURSOR** statement that are not supported in SAP Sybase IQ.
 - In the SAP Sybase IQ dialect, **DECLARE CURSOR** in a procedure or batch must immediately follow the BEGIN keyword. In the Transact-SQL dialect, there is no such restriction.
 - In Adaptive Server Enterprise, when a cursor is declared in a procedure or batch, it exists for the duration of the procedure or batch. In SAP Sybase IQ, if a cursor is declared inside a compound statement, it exists only for the duration of that compound statement (whether it is declared in an SAP Sybase IQ or Transact-SQL compound statement).

Permissions

None

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240

DECLARE LOCAL TEMPORARY TABLE Statement

Declares a local temporary table.

Syntax

```

DECLARE LOCAL TEMPORARY TABLE table-name
... ( column-definition [ column-constraint ] ...
[ , column-definition [ column-constraint ] ... ]
[ , table-constraint ] ... )
...[ ON COMMIT { DELETE | PRESERVE } ROWS
]

```

Examples

- **Example 1** – declares a local temporary table in Embedded SQL:

```

EXEC SQL DECLARE LOCAL TEMPORARY TABLE MyTable (
    number INT
);

```

- **Example 2** – declares a local temporary table in a stored procedure:

```

BEGIN
    DECLARE LOCAL TEMPORARY TABLE TempTab (
        number INT
    );

```



```
...
END
```

Usage

DECLARE LOCAL TEMPORARY TABLE declares a temporary table.

A local temporary table and the rows in it are visible only to the connection that created the table and inserted the rows. By default, the rows of a temporary table are deleted on **COMMIT**.

Declared local temporary tables within compound statements exist within the compound statement. Otherwise, the declared local temporary table exists until the end of the connection.

See *CREATE TABLE Statement* for definitions of *column-definition*, *column-constraint*, and *table-constraint* syntax. See *SELECT Statement* for an example of how to select data into a temporary table.

Once you create a local temporary table, either implicitly or explicitly, you cannot create another temporary table of that name for as long as the temporary table exists. For example, you can create a local temporary table implicitly:

```
select * into #tmp from table1
```

or you can create a local temporary table with an explicit declaration:

```
declare local temporary table foo
```

Then if you try to select into #tmp or foo, or declare #tmp or foo again, you receive an error indicating that #tmp or foo already exists.

When you declare a local temporary table, omit the owner specification. If you specify the same `owner.table` in more than one **DECLARE LOCAL TEMPORARY TABLE** statement in the same session, a syntax error is reported. For example, an error is reported when these statements are executed in the same session:

```
DECLARE LOCAL TEMPORARY TABLE user1.temp(coll int);
DECLARE LOCAL TEMPORARY TABLE user1.temp(coll int);
```

If the owner name is omitted, then the error `Item temp already exists` is reported:

```
DECLARE LOCAL TEMPORARY TABLE temp(coll int);
DECLARE LOCAL TEMPORARY TABLE temp(coll int);
```

An attempt to create a base table or a global temporary table fails, if a local temporary table of the same name exists on that connection, as the new table cannot be uniquely identified by *owner.table*.

You can, however, create a local temporary table with the same name as an existing base table or global temporary table. References to the table name access the local temporary table, as local temporary tables are resolved first.

For example, consider this sequence:

SQL Statements

```
CREATE TABLE t1 (c1 int);
INSERT t1 VALUES (9);

DECLARE LOCAL TEMPORARY TABLE t1 (c1 int);
INSERT t1 VALUES (8);

SELECT * FROM t1;
```

The result returned is 8. Any reference to `t1` refers to the local temporary table `t1` until the local temporary table is dropped by the connection.

You cannot use the **ALTER TABLE** and **DROP INDEX** statements on local temporary tables.

You cannot use the **sp_iqindex**, **sp_iqtablesiz**, and **sp_iqindexsize** stored procedures on local temporary tables.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Adaptive Server Enterprise does not support **DECLARE TEMPORARY TABLE**.

Permissions

None

See also

- *CREATE TABLE Statement* on page 205
- *SELECT Statement* on page 417

DELETE Statement

Deletes all the rows from the named table that satisfy the search condition. If no **WHERE** clause is specified, all rows from the named table are deleted.

Syntax

```
DELETE [ FROM ] [ owner.]table-name
...[ FROM table-list ]
...[ WHERE search-condition ]
```

Examples

- **Example 1** – removes employee 105 from the database:

```
DELETE
FROM Employees
WHERE EmployeeID = 105
```

- **Example 2** – removes all data prior to 1993 from the `FinancialData` table:

```
DELETE
FROM FinancialData
WHERE Year < 1993
```

- **Example 3** – removes all names from the `Contacts` table if they are already present in the `Customers` table:

```
DELETE
FROM Contacts
FROM Contacts, Customers
WHERE Contacts.Surname = Customers.Surname
AND Contacts.GivenName = Customers.GivenName
```

Usage

DELETE can be used on views provided the **SELECT** statement defining the view has only one table in the **FROM** clause and does not contain a **GROUP BY** clause, an aggregate function, or involve a **UNION** operation.

The optional second **FROM** clause in the **DELETE** statement allows rows to be deleted based on joins. If the second **FROM** clause is present, the **WHERE** clause qualifies the rows of this second **FROM** clause. Rows are deleted from the table name given in the first **FROM** clause.

Note: You cannot use the **DELETE** statement on a join virtual table. If you attempt to delete from a join virtual table, an error is reported.

There is a potential ambiguity in table names in **DELETE** statements with two **FROM** clauses that use correlation names, as illustrated in the following statement.

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_2 AS alias_2
WHERE ...
```

The table `table_1` is identified without a correlation name in the first **FROM** clause, but with a correlation name in the second **FROM** clause. In this case, `table_1` in the first clause is identified with `alias_1` in the second clause. In this illustration, there is only one instance of `table_1`.

This is an exception to the general rule that where the same table is identified with and without a correlation name in the same statement, two instances of the table are considered.

Consider this example:

```
DELETE
FROM table_1
FROM table_1 AS alias_1, table_1 AS alias_2
WHERE ...
```

In this case, there are two instances of `table_1` in the second **FROM** clause. There is no way of identifying which instance the first **FROM** clause should be identified with. The usual rules

of correlation names apply, and `table_1` in the first FROM clause is identified with neither instance in the second clause: there are three instances of `table_1` in the statement.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise, including the vendor extension.

Permissions

Requires DELETE privilege on the table.

See also

- *FROM Clause* on page 295
- *INSERT Statement* on page 327
- *TRUNCATE Statement* on page 444

DELETE (positioned) Statement [ESQL] [SP]

Deletes the data at the current location of a cursor.

Syntax

```
DELETE [ FROM table-spec ]
      WHERE CURRENT OF cursor-name
```

```
cursor-name:
  identifier | hostvar
```

```
table-spec:
  [ owner. ] correlation-name
```

Parameters

- **FROM** – the table FROM which rows are deleted is determined as follows:
 - If no FROM clause is included, the cursor can only be on a single table.
 - If the cursor is for a joined query (including using a view containing a join), you must use the FROM clause. Only the current row of the specified table is deleted. The other tables involved in the join are not affected.
 - If you include a FROM clause and do not specify table owner, table-spec is first matched against any correlation names.
 - If a correlation name exists, table-spec is identified with the correlation name.
 - If a correlation name does not exist, table-spec must be unambiguously identifiable as a table name in the cursor.

- If a FROM clause is included, and a table owner is specified, table-spec must be unambiguously identifiable as a table name in the cursor.

Examples

- **Example 1** – removes the current row from the database:

```
DELETE WHERE CURRENT OF cur_employee
```

Usage

This form of the **DELETE** statement deletes the current row of the specified cursor. The current row is defined to be the last row fetched from the cursor.

The positioned **DELETE** statement can be used on a cursor open on a view as long as the view is updatable.

Changes effected by positioned **DELETE** statements are visible in the cursor result set, except where client-side caching prevents seeing these changes.

Standards

- SQL—The range of cursors that can be updated may contain vendor extensions to ISO/ANSI SQL grammar if the `ANSI_UPDATE_CONSTRAINTS` option is set to OFF.
- Sybase—Embedded SQL use is supported by Open Client/Open Server. Procedure and trigger use is supported in SQL Anywhere.

Permissions

Requires DELETE privilege on tables used in the cursor.

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *INSERT Statement* on page 327
- *UPDATE Statement* on page 448
- *UPDATE (positioned) Statement [ESQL] [SP]* on page 451

DESCRIBE Statement [ESQL]

Gets information about the host variables required to store data retrieved from the database or host variables used to pass data to the database.

Syntax

```
DESCRIBE  
... [ USER TYPES ]
```

```

...[ { ALL | BIND VARIABLES FOR | INPUT
| OUTPUT | SELECT LIST FOR } ]
...[ { LONG NAMES [ long-name-spec ] | WITH VARIABLE RESULT } ]
...[ FOR ] { statement-name | CURSOR cursor-name }
...INTO sqlda-name

```

```

long-name-spec:
  { OWNER.TABLE.COLUMN
  | TABLE.COLUMN
  | COLUMN }

```

Parameters

- **statement-name** – identifier or host-variable. If you specify a statement name, the statement must have been previously prepared using the **PREPARE** statement with the same statement name and the SQLDA must have been previously allocated (see *ALLOCATE DESCRIPTOR Statement [ESQL]*).
- **cursor-name** – declared cursor. The cursor must have been previously declared and opened. The default action is to describe the OUTPUT. Only **SELECT** statements and **CALL** statements have OUTPUT. A DESCRIBE OUTPUT on any other statement, or on a cursor that is not a dynamic cursor, indicates no output by setting the sqld field of the SQLDA to zero.
- **sqlda-name** – identifier
- **USER TYPES** – returns information about user-defined data types of a column. Typically, such a **DESCRIBE** is done when a previous **DESCRIBE** returns an indicator of DT_HAS_USERTYPE_INFO.

The information returned is the same as for a **DESCRIBE** without the USER TYPES clause, except that the sqlname field holds the name of the user-defined data type, instead of the name of the column.

If **DESCRIBE** uses the LONG NAMES clause, the sqldata field holds this information.

- **ALL** – describes INPUT and OUTPUT with one request to the database server. This has a performance benefit in a multiuser environment. The INPUT information is filled in the SQLDA first, followed by the OUTPUT information. The sqld field contains the total number of INPUT and OUTPUT variables. The DT_DESCRIBE_INPUT bit in the indicator variable is set for INPUT variables and clear for OUTPUT variables.
- **BIND VARIABLES FOR** – equivalent to the INPUT clause. When used with the INPUT clause, DESCRIBE BIND VARIABLES does not set up the data types in the SQLDA: this needs to be done by the application.
- **SELECT LIST FOR** – equivalent to the OUTPUT clause.
- **INPUT** – fills in the name fields in the SQLDA with the bind variable names. A bind variable is a value supplied by the application when the database executes the statements. Bind variables can be considered parameters to the statement. INPUT clause also puts the number of bind variables in the sqld field of the SQLDA.

DESCRIBE uses the indicator variables in the SQLDA to provide additional information. DT_PROCEDURE_IN and DT_PROCEDURE_OUT are bits that are set in the indicator variable when a **CALL** statement is described. DT_PROCEDURE_IN indicates an IN or INOUT parameter and DT_PROCEDURE_OUT indicates an INOUT or OUT parameter. Procedure RESULT columns has both bits clear. After a DESCRIBE OUTPUT, these bits can be used to distinguish between statements that have result sets (need to use **OPEN, FETCH, RESUME, CLOSE**) and statements that do not (need to use **EXECUTE**). DESCRIBE INPUT sets DT_PROCEDURE_IN and DT_PROCEDURE_OUT appropriately only when a bind variable is an argument to a **CALL** statement; bind variables within an expression that is an argument in a **CALL** statement sets the bits.

- **OUTPUT** – fills in the data type and length in the SQLDA for each select list item. The name field is also filled in with a name for the select list item. If an alias is specified for a select list item, the name is that alias. Otherwise, the name derives from the select list item: if the item is a simple column name, it is used; otherwise, a substring of the expression is used. **DESCRIBE** also puts the number of select list items in the sqld field of the SQLDA.
 - If the statement being described is a UNION of two or more **SELECT** statements, the column names returned for DESCRIBE OUTPUT are the same column names which would be returned for the first **SELECT** statement.
 - If you describe a CALL statement, DESCRIBE OUTPUT fills in the data type, length, and name in the SQLDA for each INOUT or OUT parameter in the procedure. DESCRIBE OUTPUT also puts the number of INOUT or OUT parameters in the sqld field of the SQLDA.
 - If you describe a CALL statement with a result set, OUTPUT fills in the data type, length, and name in the SQLDA for each RESULT column in the procedure definition. DESCRIBE OUTPUT also puts the number of result columns in the sqld field of the SQLDA.
- **LONG NAMES** – retrieves column names for a statement or cursor. Without this clause, there is a 29-character limit on the length of column names: with the clause, names of an arbitrary length are supported. If LONG NAMES is used, the long names are placed into the SQLDATA field of the SQLDA, as if you were fetching from a cursor. None of the other fields (SQLLEN, SQLTYPE, and so on) are filled in. The SQLDA must be set up like a **FETCH** SQLDA: it must contain one entry for each column, and the entry must be a string type. The default specification for the long names is TABLE.COLUMN.
- **WITH VARIABLE RESULT** – describes procedures that might have more than one result set, with different numbers or types of columns. If WITH VARIABLE RESULT is used, the database server sets the SQLCOUNT value after the describe to one of these values:
 - **0** – the result set may change: the procedure call should be described again following each **OPEN** statement.
 - **1** – the result set is fixed. No redescrbing is required.

Examples

- **Example 1** – how to use the **DESCRIBE** statement:

```
sqllda = alloc_sqllda( 3 );
EXEC SQL DESCRIBE OUTPUT
    FOR employee_statement
    INTO sqllda;
if( sqllda->sqlld > sqllda->sqln ) {
    actual_size = sqllda->sqlld;
    free_sqllda( sqllda );
    sqllda = alloc_sqllda( actual_size );
    EXEC SQL DESCRIBE OUTPUT
        FOR employee_statement
        INTO sqllda;
}
```

Usage

DESCRIBE sets up the named SQLDA to describe either the OUTPUT (equivalently **SELECT LIST**) or the INPUT (BIND VARIABLES) for the named statement.

Standards

- SQL— Some clauses are vendor extensions to ISO/ANSI SQL grammar.
- Sybase—Some clauses supported by Open Client/Open Server.

Permissions

None

See also

- *ALLOCATE DESCRIPTOR Statement [ESQL]* on page 5
- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *OPEN Statement [ESQL] [SP]* on page 369
- *PREPARE Statement [ESQL]* on page 375

DISCONNECT Statement [Interactive SQL]

Drops a connection with the database.

Syntax

```
DISCONNECT [ { connection-name | CURRENT | ALL } ]
```


Parameters

- **connection-name** – identifier, string, or host-variable
- **ALL** – drops all of the connections of the application to all database environments.
- **CURRENT** – (default) Drops the current connection.

Examples

- **Example 1** – how to use **DISCONNECT** in Embedded SQL:

```
EXEC SQL DISCONNECT :conn_name
```

- **Example 2** – how to use **DISCONNECT** from **dbisql** to disconnect all connections:

```
DISCONNECT ALL
```

Usage

The **DISCONNECT** statement drops a connection with the database server and releases all resources used by it. If the connection to be dropped was named on the **CONNECT** statement, then the name can be specified.

An implicit **ROLLBACK** is executed on connections that are dropped.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *CONNECT Statement [ESQL] [Interactive SQL]* on page 104
- *SET CONNECTION Statement [ESQL] [Interactive SQL]* on page 429

DROP Statement

Removes objects from the database.

Syntax

```
DROP
  { DBSPACE dbspace-name
  | { DATATYPE [ IF EXISTS ]
  | DOMAIN [ IF EXISTS ] } datatype-name
```

```

| EVENT [ IF EXISTS ] event-name
| INDEX [ IF EXISTS ] [ [ owner ]. table-name . ] index-name
| MESSAGE message-number
| TABLE [ IF EXISTS ] [ owner . ] table-name
| VIEW [ IF EXISTS ] [ owner . ] view-name
| MATERIALIZED VIEW [ IF EXISTS ] [ owner . ] view-name
| PROCEDURE [ IF EXISTS ] [ owner . ] procedure-name
| FUNCTION [ IF EXISTS ] [ owner . ] function-name }

```

Parameters

- **IF EXISTS** – use if you do not want an error returned when the **DROP** statement attempts to remove a database object that does not exist.
- **INDEX** – deletes any explicitly created index. It deletes an implicitly created index only if there are no unique or foreign-key constraints or associated primary key.

DROP INDEX for a nonunique HG index fails if an associated unenforced foreign key exists.

Warning! Do not delete views owned by the DBO user. Deleting such views or changing them into tables might cause problems.

DROP INDEX is prevented whenever the statement affects a table that is currently being used by another connection.

- **TABLE – DROP TABLE** is prevented if the primary table has foreign-key constraints associated with it, including unenforced foreign-key constraints

DROP TABLE is also prevented if the table has an **IDENTITY** column and **IDENTITY_INSERT** is set to that table. To drop the table you must clear **IDENTITY_INSERT**, that is, set **IDENTITY_INSERT** to '' (an empty string), or set to another table name.

A foreign key can have either a nonunique single or a multicolumn HG index. A primary key may have unique single or multicolumn HG indexes. You cannot drop the HG index implicitly created for an existing foreign key, primary key, and unique constraint.

The four initial dbspaces are **SYSTEM**, **IQ_SYSTEM_MAIN**, **IQ_SYSTEM_TEMP**, and **IQ_SYSTEM_MSG**. You cannot drop these initial dbspaces, but you may drop dbspaces from the IQ main store or catalog store, which may contain multiple dbspaces, as long as at least one dbspace remains with readwrite mode.

You must drop tables in the dbspace before you can drop the dbspace. An error is returned if the dbspace still contains user data; other structures are automatically relocated when the dbspace is dropped. You can drop a dbspace only after you make it read-only.

Note: A dbspace may contain data at any point after it is used by a command, thereby preventing a **DROP DBSPACE** on it.

DROP TABLE is prevented whenever the statement affects a table that is currently being used by another connection.

- **PROCEDURE – DROP PROCEDURE** is prevented when the procedure is in use by another connection.
- **DATATYPE – DROP DATATYPE** is prevented if the data type is used in a table. You must change data types on all columns defined on the user-defined data type to drop the data type. It is recommended that you use **DROP DOMAIN** rather than **DROP DATATYPE**, as **DROP DOMAIN** is the syntax used in the ANSI/ISO SQL3 draft.

Examples

- **Example 1** – drop the `Departments` table from the database:

```
DROP TABLE Departments
```

- **Example 2** – drop the `emp_dept` view from the database:

```
DROP VIEW emp_dept
```

Usage

DROP removes the definition of the indicated database structure. If the structure is a `dbspace`, then all tables with any data in that `dbspace` must be dropped or relocated prior to dropping the `dbspace`; other structures are automatically relocated. If the structure is a table, all data in the table is automatically deleted as part of the dropping process. Also, all indexes and keys for the table are dropped by **DROP TABLE**.

DROP DBSPACE is prevented whenever the statement affects a table that is currently being used by another connection. **DROP PROCEDURE** is prevented when the procedure is in use by another connection.

Side Effects

- Automatic commit. Clears the Data window in **dbisql**. **DROP TABLE** and **DROP INDEX** close all cursors for the current connection.
- Local temporary tables are an exception; no commit is performed when one is dropped.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

DBSPACE clause – Requires the `DROP ANY OBJECT` system privilege and user must be the only connection to the database.

DOMAIN clause – Requires one of:

SQL Statements

- DROP DATATYPE system privilege.
- DROP ANY OBJECT system privilege.
- You own the object.

FUNCTION clause – Requires one of:

- DROP ANY PROCEDURE system privilege.
- DROP ANY OBJECT system privilege.
- You own the function.

INDEX clause – Requires one of:

- DROP ANY INDEX system privilege.
- DROP ANY OBJECT system privilege.
- REFERENCE privilege on the underlying table being indexed.
- You own the underlying table being indexed.

DBA or users with the appropriate privilege can drop an index on tables that are owned other users without using a fully-qualified name. All other users must provide a fully-qualified index name to drop an index on a base table owned by the DBA.

MATERIALIZED VIEW clause – Requires one of:

- DROP ANY MATERIALIZED VIEW system privilege.
- DROP ANY OBJECT system privilege.
- You own the materialized view.

PROCEDURE clause – Requires one of:

- DROP ANY PROCEDURE system privilege.
- DROP ANY OBJECT system privilege.
- You own the procedure.

TABLES clause – Requires one of:

- DROP ANY TABLE system privilege.
- DROP ANY OBJECT system privilege.
- You own the table.

Global temporary tables cannot be dropped unless all users that have referenced the temporary table have disconnected.

VIEW clause – Requires one of:

- DROP ANY VIEW system privilege.
- DROP ANY OBJECT system privilege.
- You own the view.

All other clauses – Requires one of:

- DROP ANY OBJECT system privilege.
- You own the object.

DROP CONNECTION Statement

Drops any user connection to the database.

Syntax

```
DROP CONNECTION connection-id
```

Parameters

- **connection-id** – obtained using the **connection_property** function to request the connection number. This statement returns the connection ID of the current connection:

```
SELECT connection_property( 'number' )
```

Examples

- **Example 1** – drop connection with ID number 4:

```
DROP CONNECTION 4
```

Usage

You cannot drop your current connection; you must first create another connection, then drop your first connection.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the DROP CONNECTION system privilege.

See also

- *CONNECT Statement [ESQL] [Interactive SQL]* on page 104

DROP DATABASE Statement

Drops a database and its associated dbspace segment files.

Syntax

```
DROP DATABASE db-filename [ KEY key-spec ]
```

Parameters

- **key-spec** – a string, including mixed cases, numbers, letters, and special characters. It might be necessary to protect the key from interpretation or alteration by the command shell.
- **db-filename** – corresponds to the database file name you defined for the database using **CREATE DATABASE**. If you specified a directory path for this value in the **CREATE DATABASE** command, you must also specify the directory path for **DROP DATABASE**. Otherwise, SAP Sybase IQ looks for the database files in the default directory where the server files reside.

Examples

- **Example 1** – drops database mydb:

```
DROP DATABASE 'mydb.db'
```

- **Example 2** – drops the encrypted database marvin.db, which was created with the key is!seCret:

```
DROP DATABASE 'marvin.db' KEY 'is!seCret'
```

- **Example 3** – drops the database temp.db from the /s1/temp directory on a UNIX system:

```
DROP DATABASE '/s1/temp/temp.db'
```

Usage

DROP DATABASE drops all the database segment files associated with the IQ store and temporary store before it drops the catalog store files.

You must stop a database before you can drop it. If the connection parameter AUTOSTOP=no is used, you may need to issue a **STOP DATABASE** statement.

You cannot execute a **DROP DATABASE** statement to drop an IQ database that has a DatabaseStart event defined for it.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

The permissions required to execute this statement are set using the **-gu** server command line option, as follows:

- **NONE** – No user can issue this statement.
- **DBA** – Requires the SERVER OPERATOR system privilege.
- **UTILITY_DB** – Only those users who can connect to the `utility_db` database can issue this statement.

See also

- *CREATE DATABASE Statement* on page 107
- *STOP DATABASE Statement [Interactive SQL]* on page 441

DROP EXTERNLOGIN Statement

Drops an external login from the SAP Sybase IQ system tables.

Syntax

```
DROP EXTERNLOGIN login-name
  TO remote-server
```

Parameters

- **login-name** – specifies the local user login name.
- **TO** – specifies the name of the remote server. The alternate login name of the local user and password for that server is the external login that is deleted.

Examples

- **Example 1** – drops the login `dba` from the remote database `sybase1`:

```
DROP EXTERNLOGIN dba TO sybase1
```

Usage

Changes made by **DROP EXTERNLOGIN** do not take effect until the next connection to the remote server.

SQL Statements

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *CREATE EXTERNLOGIN Statement* on page 131

DROP LDAP SERVER Statement

Removes the named LDAP server configuration object from the `SYSLDAPSERVER` system view after verifying that the LDAP server configuration object is not in a `READY` or `ACTIVE` state.

Syntax

```
DROP LDAP SERVER ldapua-server-name  
[ WITH DROP ALL REFERENCES ] [ WITH SUSPEND ]
```

Parameters

- **WITH DROP ALL REFERENCES** – allows the removal of an LDAP server configuration object from service that has a reference in a login policy.
- **WITH SUSPEND** – allows an LDAP server configuration object to be dropped even if in a `READY` or `ACTIVE` state.

Examples

- **Example 1** – assuming that references to the LDAP server configuration object have been removed from all login policies, the following two sets of commands are equivalent. Using the `WITH DROP ALL REFERENCES` and `WITH SUSPEND` parameters eliminates the need to execute an **ALTER LDAP SERVER** statement before the **DROP LDAP SERVER** statement:

```
DROP LDAP SERVER ldapserver1 WITH DROP ALL REFERENCES WITH SUSPEND
```

is equivalent to


```
ALTER LDAP SERVER ldapserver1 WITH SUSPEND DROP LDAP SERVER
ldapserver1 WITH DROP ALL REFERENCES
```

Usage

The **DROP LDAP SERVER** statement fails when it is issued against an LDAP server configuration object that is in a **READY** or **ACTIVE** state. This ensures that an LDAP server configuration object in active use cannot be accidentally dropped. The **DROP LDAP SERVER** statement also fails if a login policy exists with a reference to the LDAP server configuration object.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the **MANAGE ANY LDAP SERVER** system privilege.

DROP LOGIN POLICY Statement

Removes a login policy from the database.

Syntax

```
DROP LOGIN POLICY policy-name
```

Examples

- **Example 1** – create and then delete the `Test11` login policy:

```
CREATE LOGIN POLICY Test11;
DROP LOGIN POLICY Test11 ;
```

Usage

A **DROP LOGIN POLICY** statement fails if you attempt to drop a policy that is assigned to a user. You can use either the **ALTER USER** statement to change the policy assignment of the user or **DROP USER** to drop the user.

Permissions

Requires the **MANAGE ANY LOGIN POLICY** system privilege.

See also

- *ALTER USER Statement* on page 71
- *CREATE LOGIN POLICY Statement* on page 155

- *DROP USER Statement* on page 275
- *ALTER LOGIN POLICY Statement* on page 24

DROP LOGICAL SERVER Statement

Drops a user-defined logical server. This statement enforces consistent shared system temporary store settings across physical nodes shared by logical servers.

Syntax

```
DROP LOGICAL SERVER logical-server-name  
[ WITH STOP SERVER ]
```

Parameters

- **WITH STOP SERVER** – automatically shuts down all servers in the logical server when the `TEMP_DATA_IN_SHARED_TEMP` option is changed directly or indirectly.

Applies to

Multiplex only.

Examples

- **Example 1** – drops a user-defined logical server `ls1`:

```
DROP LOGICAL SERVER ls1
```

Usage

SAP Sybase IQ performs the following catalog changes internally when dropping a logical server:

- Drops all membership definitions of the logical server.
- Drops its logical server assignment from each login policy that has an explicit assignment to the subject logical server. If it is the only logical server assigned to the login policy, SAP Sybase IQ sets the logical server assignment for the login policy to `NONE`.
- Removes the logical server entry from `ISYSIQ.LOGICALSERVER`.

Permissions

Requires the `MANAGE MULTIPLEX` system privilege.

DROP LS POLICY Statement

Removes a logical server policy from the multiplex.

Syntax

```
DROP LS POLICY ls-policy-name
```

Parameters

- **ls-policy-name** – any policy name except ROOT and must refer to a policy not currently used for any logical server.

Applies to

Multiplex.

Examples

- **Example 1** – deletes the Test20 logical server policy:

```
DROP LS POLICY Test20
```

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE MULTIPLEX system privilege.

DROP MULTIPLEX SERVER Statement

Deletes a server from the multiplex.

Syntax

```
DROP MULTIPLEX SERVER {server-name} [drop_mpx_server_clause]
```

```
drop_mpx_server_clause:  
{ WITH DROP MEMBERSHIP | WITH DROP LOGICAL SERVER }
```

Parameters

- **WITH DROP MEMBERSHIP** – fails with an error, when one or more logical server memberships exist for the multiplex server being dropped. Use the **WITH DROP MEMBERSHIP** clause to drop the multiplex server along with all of its memberships.
- **WITH DROP LOGICAL SERVER** – drops the last secondary server along with all user-defined logical servers. When dropping the last secondary server, the **DROP MULTIPLEX SERVER** command fails, when there are one or more user-defined logical servers.

Note: The **WITH DROP LOGICAL SERVER** clause is only valid when dropping the last secondary server. An error is reported otherwise.

Applies to

Multiplex only.

Examples

- **Example 1** – drops a multiplex server named *writer1*:

```
DROP MULTIPLEX SERVER writer1
```

Usage

Shut down each multiplex server before dropping it. This statement automatically commits.

If not already stopped as recommended, the dropped server automatically shuts down after executing this statement.

Dropping the last secondary server converts the multiplex back to simplex. After dropping the last secondary server within the multiplex, the coordinator automatically shuts down. If required, it needs to be restarted.

Permissions

Requires the **MANAGE MULTIPLEX** system privilege.

DROP ROLE Statement

Removes a user-defined role from the database or converts a user-extended role to a regular user.

Syntax

```
DROP ROLE [ FROM USER ] role_name  
[ WITH REVOKE ]
```

Parameters

- **role_name** – must be the name of a role that already exists in the database.
- **FROM USER** – required to convert a user-extended role back to act as a regular user rather than remove it from the database. The *role_name* must exist in the database.

The user retains any login privileges, system privileges, and roles granted to the user-extended role and becomes the owner of any objects owned by the user-extended role. Any users granted to the user-extended are immediately revoked.

- **WITH REVOKE** – required when dropping a standalone or user-extended role to which users have been granted the underlying system privileges of the role with either the WITH ADMIN OPTION or WITH NO ADMIN OPTION clause.

Examples

- **Example 1** – converts a user-extended role named `Joe` that has not been granted to other users or roles back to a regular user:

```
DROP ROLE FROM USER Joe
```

- **Example 2** – drops a user-extended role named `Jack` that has not been granted to other users or roles from the database:

```
DROP ROLE Jack
```

- **Example 3** – converts a user-extended role named `Sam` that has been granted to other user or roles back to a regular role:

```
DROP ROLE FROM USER Sam
WITH REVOKE
```

- **Example 4** – drops a standalone role named `Sales2` that has been granted to other users or roles from the database:

```
DROP ROLE Sales2
WITH REVOKE
```

Usage

A user-defined role can be dropped from the database or converted back to a regular user at any time as long as all dependent roles left meet the minimum required number of administrative users with active passwords.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

- Requires administrative rights over the role being dropped.
- If the role being dropped owns objects, none are in use by any user in any session at the time the DROP statement is executed.

DROP SEQUENCE statement

Drops a sequence. This statement applies to SAP Sybase IQ catalog store tables only.

Syntax

```
DROP SEQUENCE [ owner. ] sequence-name
```

Remarks

If the named sequence cannot be located, an error message is returned. When you drop a sequence, all synonyms for the name of the sequence are dropped automatically by the database server.

Privileges

You must be the owner of the sequence, or have the DROP ANY SEQUENCE or DROP ANY OBJECT system privilege.

Side effects

None

Standards and compatibility

- **SQL/2008** – Sequences comprise SQL/2008 optional language feature T176.

Example

The following example creates and then drops a sequence named Test:

```
CREATE SEQUENCE Test
START WITH 4
INCREMENT BY 2
NO MAXVALUE
NO CYCLE
CACHE 15;
DROP SEQUENCE Test;
```

DROP SERVER Statement

Drops a remote server from the SAP Sybase IQ system tables.

Syntax

```
DROP SERVER server-name
```

Examples

- **Example 1** – this example drops the server IQ_prod:

```
DROP SERVER iq_prod
```

Usage

Before **DROP SERVER** succeeds, you must drop all the proxy tables that have been defined for the remote server.

Side Effects

- Automatic commit

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

Requires the SERVER OPERATOR system privilege.

See also

- *CREATE SERVER Statement* on page 192

DROP SERVICE Statement

Deletes a Web service.

Syntax

```
DROP SERVICE service-name
```

Examples

- **Example 1** – drops a Web service named `tables`:

```
DROP SERVICE tables
```

Usage

DROP SERVICE deletes a Web service.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

Requires the **MANAGE ANY WEB SERVICE** system privilege.

See also

- *ALTER SERVICE Statement* on page 44
- *CREATE SERVICE Statement* on page 194

DROP SPATIAL REFERENCE SYSTEM Statement

Drops a spatial reference system.

Syntax

```
DROP SPATIAL REFERENCE SYSTEM [ IF EXISTS ] name
```

Parameters

- **IF EXISTS** – prevents an error from being returned when the **DROP SPATIAL REFERENCE SYSTEM** statement attempts to remove a spatial reference system that does not exist.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- **MANAGE ANY SPATIAL OBJECT** system privilege.

- DROP ANY OBJECT system privilege.
- You own the spatial references system

DROP SPATIAL UNIT OF MEASURE Statement

Drops a spatial unit of measurement.

Syntax

DROP SPATIAL UNIT OF MEASURE [IF EXISTS] *identifier*

Parameters

- **IF EXISTS** – prevents an error from being returned when the **DROP SPATIAL UNIT OF MEASURE** statement attempts to remove a spatial unit of measure that does not exist.

Examples

- **Example** – the following example drops a fictitious spatial unit of measure named Test:

```
DROP SPATIAL UNIT OF MEASURE Test;
```

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- MANAGE ANY SPATIAL OBJECT system privilege.
- DROP ANY OBJECT system privilege.
- You own the spatial unit of measure

DROP STATEMENT Statement [ESQL]

Frees resources used by the named prepared statement. These resources are allocated by a successful **PREPARE** statement, and are normally not freed until the database connection is released.

Syntax

DROP STATEMENT [*owner.*] *statement-name*

Parameters

- **statement-name** – identifier or host-variable

Examples

- **Example 1** – drops the statements `s1` and `stmt`:

```
EXEC SQL DROP STATEMENT s1;  
EXEC SQL DROP STATEMENT :stmt;
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported in Open Client/Open Server

Permissions

Must have prepared the statement.

See also

- *PREPARE Statement [ESQL]* on page 375

DROP TEXT CONFIGURATION Statement

Drops a text configuration object.

Note: This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Syntax

```
DROP TEXT CONFIGURATION [ owner. ] text-config-name
```

Examples

- **Example 1** – creates and drops the `mytextconfig` text configuration object:

```
CREATE TEXT CONFIGURATION mytextconfig FROM default_char;  
DROP TEXT CONFIGURATION mytextconfig;
```

Usage

Use **DROP TEXT CONFIGURATION** to drop a text configuration object.

Attempting to drop a text configuration object with dependent TEXT indexes results in an error. You must drop the dependent TEXT indexes before dropping the text configuration object.

Text configuration objects are stored in the `ISYSTETEXTCONFIG` system table.

Side Effects:

- Automatic commit.

Permissions

Text configuration object owned by self – None required.

Table configuration object owned by any user – Requires one of:

- DROP ANY TEXT CONFIGURATION system privilege.
- DROP ANY OBJECT system privilege.

DROP TEXT INDEX Statement

Removes a TEXT index from the database.

Note: This statement requires the Unstructured Data Analytics (IQ_UDA) license.

Syntax

```
DROP TEXT INDEX text-index-name
ON [ owner ] table-name
```

Parameters

- **ON** – specifies the table on which the TEXT index is built.

Examples

- **Example 1** – creates and drops the TextIdx TEXT index:

```
CREATE TEXT INDEX TextIdx ON Customers ( Street );
DROP TEXT INDEX TextIdx ON Customers;
```

Usage

You must drop dependent TEXT indexes before you can drop a text configuration object.

Side Effects:

- Automatic commit.

Permissions

Requires one of:

- DROP ANY INDEX system privilege.
- DROP ANY OBJECT system privilege.

- REFERENCE privilege on the table being indexed.
- You own the underlying table.

DROP TRIGGER statement

Removes a trigger from the database.

Syntax

```
DROP TRIGGER [ IF EXISTS ] [ owner. ] [ table-name. ] trigger-name
```

Remarks

Use the IF EXISTS clause if you do not want an error returned when the DROP statement attempts to remove a database object that does not exist.

Privileges

To drop a trigger on a table, one of the following must be true:

- You are the owner of the table.
- You have ALTER privilege on the table.
- You have the ALTER ANY TABLE system privilege.
- You have the ALTER ANY OBJECT system privilege.

To drop a trigger on a view owned by someone else, you must have either the ALTER ANY VIEW or ALTER ANY OBJECT system privilege.

Side effects

Automatic commit. Clears the **Results** tab in the **Results** pane in Interactive SQL.

Standards and compatibility

- **SQL/2008** – DROP TRIGGER comprises part of optional SQL language feature T211, "Basic trigger capability", of the SQL/2008 standard. The IF EXISTS clause is a vendor extension.

Example

This example creates, and then drops, a trigger called emp_upper_postal_code to ensure that postal codes are in upper case before updating the Employees table. If the trigger does not exist, an error is returned.

```
CREATE TRIGGER emp_upper_postal_code
BEFORE UPDATE OF PostalCode
ON GROUPO.Employees
REFERENCING NEW AS new_emp
FOR EACH ROW
WHEN ( ISNUMERIC( new_emp.PostalCode ) = 0 )
BEGIN
  -- Ensure postal code is uppercase (employee might be
```

```
-- in Canada where postal codes contain letters)
SET new_emp.PostalCode = UPPER(new_emp.PostalCode)
END;
DROP TRIGGER MyTrigger;
```

DROP USER Statement

Removes a user.

Syntax

```
DROP USER user-name
```

Parameters

- **user-name** – name of the user to remove.

Examples

- **Example 1** – drops the user SQLTester from the database:

```
DROP USER SQLTester
```

Standards

- SQL – ISO/ANSI SQL compliant.
- Sybase – Not supported by Adaptive Server Enterprise.

Permissions

Requires the MANAGE ANY USER system privilege.

Note: When dropping a user, any objects owned by this user and any permissions granted by this user will be removed.

See also

- *CREATE LOGIN POLICY Statement* on page 155
- *CREATE USER Statement* on page 230
- *DROP LOGIN POLICY Statement* on page 263
- *ALTER LOGIN POLICY Statement* on page 24
- *GRANT ROLE Statement* on page 312
- *GRANT System Privilege Statement* on page 319
- *REVOKE System Privilege Statement* on page 408
- *REVOKE ROLE Statement* on page 404

DROP VARIABLE Statement

Eliminates a SQL variable that was created using the **CREATE VARIABLE** statement.

Syntax

```
DROP VARIABLE [ IF EXISTS ] identifier
```

Parameters

- **IF EXISTS** – prevents an error being returned when the **DROP VARIABLE** statement attempts to remove a database object that does not exist.

Usage

Variables are automatically eliminated when the database connection is released. Variables are often used for large objects, so eliminating them after use or setting them to NULL can free up significant resources (primarily disk space).

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

None

See also

- *CREATE VARIABLE Statement* on page 232
- *SET Statement [ESQL]* on page 426

EXECUTE Statement [ESQL]

Executes a SQL statement.

Syntax

Syntax 1 – Executes the named dynamic statement that was previously prepared

```
EXECUTE statement-name
... [ { USING DESCRIPTOR sqllda-name | USING host-variable-list } ]
... [ { INTO DESCRIPTOR into-sqllda-name | INTO into-host-variable-list } ]
... [ ARRAY :nnn ] ]
```

Syntax 2 – A short form to **PREPARE** and **EXECUTE** a statement that does not contain bind variables or output

```
EXECUTE IMMEDIATE statement
```

Parameters

- **statement-name** – identifier or host-variable
- **sqlda-name** – identifier
- **into-sqlda-name** – identifier
- **statement** – string or host-variable
- **USING** – OUTPUT from a **SELECT** statement or a **CALL** statement is put either into the variables in the variable list or into the program data areas described by the named SQLDA. The correspondence is one to one from the OUTPUT (selection list or parameters) to either the host variable list or the SQLDA descriptor array.
- **INTO** – if used with an **INSERT** statement, the inserted row is returned in the second descriptor. For example, when using autoincrement primary keys that generate primary-key values, **EXECUTE** provides a mechanism to refetch the row immediately and determine the primary-key value assigned to the row.
- **ARRAY** – used with prepared **INSERT** statements to allow wide inserts, which insert more than one row at a time and which might improve performance. The value *nnn* is the number of rows to be inserted. The SQLDA must contain *nnn* * (columns per row) variables. The first row is placed in SQLDA variables 0 to (columns per row)-1, and so on.

Examples

- **Example 1** – executes a **DELETE**:

```
EXEC SQL EXECUTE IMMEDIATE
'DELETE FROM Employees WHERE EmployeeID = 105';
```

- **Example 2** – executes a prepared **DELETE** statement:

```
EXEC SQL PREPARE del_stmt FROM
'DELETE FROM Employees WHERE EmployeeID = :a';
EXEC SQL EXECUTE del_stmt USING :employee_number;
```

- **Example 3** – executes a prepared query:

```
EXEC SQL PREPARE sell FROM
'SELECT Surname FROM Employees WHERE EmployeeID = :a';
EXEC SQL EXECUTE sell USING :employee_number INTO :emp_lname;
```

Usage

Syntax 1 — If the dynamic statement contains host variable placeholders which supply information for the request (bind variables), then either the *sqlda-name* must specify a C

SQL Statements

variable which is a pointer to an SQLDA containing enough descriptors for all bind variables occurring in the statement, or the bind variables must be supplied in the *host-variable-list*.

Syntax 2 — The SQL statement contained in the string or host variable is immediately executed and is dropped on completion.

EXECUTE can be used for any SQL statement that can be prepared. Cursors are used for **SELECT** statements or **CALL** statements that return many rows from the database.

Note: You cannot reference a Table UDF in an **EXECUTE** statement.

After successful execution of an **INSERT**, **UPDATE**, or **DELETE** statement, the sqlerrd[2] field of the SQLCA (SQLCOUNT) is filled in with the number of rows affected by the operation.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported in Open Client/Open Server.

Permissions

Permissions are checked on the statement being executed.

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *PREPARE Statement [ESQL]* on page 375

EXECUTE Statement [T-SQL]

Invokes a procedure, as an Adaptive Server Enterprise-compatible alternative to the **CALL** statement.

Syntax

```
EXECUTE [ @return_status = ] [owner.]procedure_name
... { [ @parameter-name = ] expression
| [ @parameter-name = ] @variable [ output ] } ,...
```

Examples

- **Example 1** – creates the procedure p1:

```
CREATE PROCEDURE p1( @var INTEGER = 54 )
AS
PRINT 'on input @var = %1! ', @var
DECLARE @intvar integer
SELECT @intvar=123
SELECT @var=@intvar
PRINT 'on exit @var = %1!', @var;
```


Execute the procedure, supplying the input value of 23 for the parameter. If you are connected from an Open Client application, **PRINT** messages are displayed on the client window. If you are connected from an ODBC or Embedded SQL application, messages display on the database server window.

```
EXECUTE p1 23
```

An alternative way of executing the procedure, which is useful if there are several parameters:

```
EXECUTE p1 @var = 23
```

Execute the procedure, using the default value for the parameter:

```
EXECUTE p1
```

Execute the procedure and store the return value in a variable for checking return status:

```
EXECUTE @status = p1 23
```

Usage

EXECUTE executes a stored procedure, optionally supplying procedure parameters and retrieving output values and return status information.

EXECUTE is implemented for Transact-SQL compatibility, but can be used in either Transact-SQL or SAP Sybase IQ batches and procedures.

Note: You cannot reference a Table UDF in an **EXECUTE** statement.

Permissions

Must be the owner of the procedure, have EXECUTE permission for the procedure, or have the EXECUTE ANY PROCEDURE system privilege.

See also

- *CALL Statement* on page 90

EXECUTE IMMEDIATE Statement [ESQL] [SP]

Extends the range of statements that can be executed from within procedures. It lets you execute dynamically prepared statements, such as statements that are constructed using the parameters passed in to a procedure.

Syntax

Syntax 1

```
EXECUTE IMMEDIATE [ execute-option ] string-expression
```

```
execute-option:
```

```
WITH QUOTES [ ON | OFF ]
```

```
| WITH ESCAPES { ON | OFF }
| WITH RESULT SET { ON | OFF }
```

Syntax 2

```
EXECUTE ( string-expression )
```

Parameters

- **WITH QUOTES [ON]** – any double quotes in the string expression are assumed to delimit an identifier. When not specified, the treatment of double quotes in the string expression depends on the current setting of the **QUOTED_IDENTIFIER** database option.

WITH QUOTES is useful when an object name that is passed into the stored procedure is used to construct the statement that is to be executed, but the name might require double quotes and the procedure might be called when **QUOTED_IDENTIFIER** is set to OFF.

See *QUOTED_IDENTIFIER Option [TSQL]*.

- **WITH ESCAPES** – causes any escape sequences (such as \n, \x, or \\) in the string expression to be ignored. For example, two consecutive backslashes remain as two backslashes, rather than being converted to a single backslash. The default setting is ON.

You can use **WITH ESCAPES OFF** for easier execution of dynamically constructed statements referencing file names that contain backslashes.

- **string-expression** – in some contexts, escape sequences in the *string-expression* are transformed before **EXECUTE IMMEDIATE** is executed. For example, compound statements are parsed before being executed, and escape sequences are transformed during this parsing, regardless of the WITH ESCAPES setting. In these contexts, **WITH ESCAPES OFF** prevents further translations from occurring. For example:

```
BEGIN
DECLARE String1 LONG VARCHAR;
DECLARE String2 LONG VARCHAR;
EXECUTE IMMEDIATE
  'SET String1 = 'One backslash: \\\\'';
  EXECUTE IMMEDIATE WITH ESCAPES OFF
  'SET String2 = 'Two backslashes: \\\\'';
SELECT String1, String2
END
```

- **WITH RESULT SET** – when specified with ON, the **EXECUTE IMMEDIATE** statement returns a result set. With this clause, the containing procedure is marked as returning a result set. If you do not include this clause, an error is reported when the procedure is called if the statement does not produce a result set.

Note: The default option is OFF, meaning that no result set is produced when the statement is executed.

Examples

- **Example 1** – creates a table, where the table name is supplied as a parameter to the procedure. The full **EXECUTE IMMEDIATE** statement must be on a single line.

```
CREATE PROCEDURE CreateTableProc(
    IN tablename char(30)
)
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE ' || tablename ||
    ' ( column1 INT PRIMARY KEY) '
END;
```

Call the procedure and create table mytable:

```
CALL CreateTableProc( 'mytable' )
```

Usage

Literal strings in the statement must be enclosed in single quotes, and must differ from any existing statement name in a **PREPARE** or **EXECUTE IMMEDIATE** statement. The statement must be on a single line.

Only global variables can be referenced in a statement executed by **EXECUTE IMMEDIATE**.

Only syntax 2 can be used inside Transact-SQL stored procedures.

Side Effects

None. However, if the statement is a data definition statement with an automatic commit as a side effect, then that commit does take place.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported in Open Client/Open Server.

Permissions

None. The statement is executed with the permissions of the owner of the procedure, not with the permissions of the user who calls the procedure.

See also

- *BEGIN ... END Statement* on page 84
- *CREATE PROCEDURE Statement* on page 165
- *QUOTED_IDENTIFIER Option [TSQL]* on page 617

EXIT Statement [Interactive SQL]

Leaves Interactive SQL.

Syntax

```
{ EXIT | QUIT | BYE } [ return-code ]

return-code:
    number | connection-variable
```

Examples

- **Example 1** – sets the Interactive SQL return value to 1 if there are any rows in table T, or to 0 if T contains no rows:

```
CREATE VARIABLE rowCount INT;
CREATE VARIABLE retcode INT;
SELECT COUNT(*) INTO rowCount FROM T;
IF( rowCount > 0 ) THEN
    SET retcode = 1;
ELSE
    SET retcode = 0;
END IF;
EXIT retcode;
```

Note: You cannot write the following the statement, because **EXIT** is an Interactive SQL statement (not a SQL statement), and you cannot include any Interactive SQL statement in other SQL block statements:

```
CREATE VARIABLE rowCount INT;
SELECT COUNT(*) INTO rowCount FROM T;
IF( rowCount > 0 ) THEN
    EXIT 1;    // <-- not allowed
ELSE
    EXIT 0;    // <-- not allowed
END IF;
```

Usage

Closes the Interactive SQL window, if you are running Interactive SQL as a windowed program, or terminates Interactive SQL altogether when run in command-prompt (batch) mode. In both cases, the database connection is also closed. Before closing the database connection, Interactive SQL automatically executes a **COMMIT** statement, if the **COMMIT_ON_EXIT** option is set to ON. If this option is set to OFF, Interactive SQL performs an implicit **ROLLBACK**. By default, the **COMMIT_ON_EXIT** option is set to ON.

The optional return code can be used in batch files to indicate success or failure of the commands in an Interactive SQL command file. The default return code is 0.

Side Effects

- Automatically performs a commit, if option `COMMIT_ON_EXIT` is set to `ON` (the default); otherwise this statement performs an implicit rollback.
- On Windows operating systems, the optional return value is available as `ERRORLEVEL`.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable by Adaptive Server Enterprise.

Permissions

None

See also

- *SET OPTION Statement* on page 431

FETCH Statement [ESQL] [SP]

Retrieves one row from the named cursor. The cursor must have been previously opened.

Syntax

FETCH

```
{ NEXT | PRIOR | FIRST | LAST
| ABSOLUTE row-count | RELATIVE row-count }
... cursor-name
... { [ INTO host-variable-list ]
| USING DESCRIPTOR sqlda-name
| INTO variable-list }
... [ PURGE ] [ BLOCK n ] [ ARRAY fetch-count ]
... INTO variable-list
... IQ CACHE row-count
```

These clauses are for use in Embedded SQL only:

- **USING DESCRIPTOR** *sqlda-name*
- **INTO** *host-variable-list*
- **PURGE**
- **BLOCK** *n*
- **ARRAY** *fetch-count*
- Use of *host-variable* in *cursor-name* and *row-count*

Parameters

- **cursor-name** – identifier or host variable
- **qlda-name** – identifier
- **host-variable-list** – may contain indicator variables
- **row-count** – number or host variable
- **fetch-count** – integer or host variable
- **INTO** – if it is not specified, then **FETCH** positions the cursor only. **OPEN** initially positions the cursor before the first row. An optional positional parameter can be specified that allows the cursor to be moved before a row is fetched.
- **NEXT** – (default) Causes the cursor to advance one row before the row is fetched.
- **PRIOR** – moves the cursor back one row before fetching.
- **ABSOLUTE** – used to go to a particular row. A zero indicates the position before the first row.

A one (1) indicates the first row, and so on. Negative numbers are used to specify an absolute position from the end of the cursor. A negative one (-1) indicates the last row of the cursor. **FIRST** is a short form for **ABSOLUTE 1**. **LAST** is a short form for **ABSOLUTE -1**.

Note: SAP Sybase IQ handles the **FIRST**, **LAST**, **ABSOLUTE**, and negative **RELATIVE** clauses less efficiently than some other DBMS products, so there is a performance impact when using them.

- **RELATIVE** – moves the cursor by a specified number of rows in either direction before fetching.

A positive number indicates moving forward and a negative number indicates moving backwards. Thus, a **NEXT** is equivalent to **RELATIVE 1** and **PRIOR** is equivalent to **RELATIVE -1**. **RELATIVE 0** retrieves the same row as the last fetch statement on this cursor.

- **BLOCK** – gives the client and server a hint as to how many rows may be fetched by the application. The special value of 0 means the request is sent to the server and a single row is returned (no row blocking).
- **PURGE** – causes the client to flush its buffers of all rows and then send the fetch request to the server. This fetch request may return a block of rows.
- **ARRAY** – allows wide fetches, which retrieve more than one row at a time, and which might improve performance. To use wide fetches in Embedded SQL, include the **FETCH** statement in your code:

```
EXEC SQL FETCH . . . ARRAY nnn
```

where ARRAY nnn is the last item of the **FETCH** statement. The fetch count nnn can be a host variable. The **SQLDA** must contain nnn * (columns per row) variables. The first row is placed in **SQLDA** variables 0 to (columns per row)-1, and so on.

- **IQ CACHE** – specifies the maximum number of rows buffered in the FIFO queue. If you do not specify a value for **IQ CACHE**, the value of the **CURSOR_WINDOW_ROWS** database option is used. The default setting of **CURSOR_WINDOW_ROWS** is 200.

Examples

- **Example 1** – embedded SQL example:

```
EXEC SQL DECLARE cur_employee CURSOR FOR
SELECT EmployeeID, Surname FROM Employees;
EXEC SQL OPEN cur_employee;
EXEC SQL FETCH cur_employee
INTO :emp_number, :emp_name:indicator;
```

- **Example 2** – procedure example:

```
BEGIN
    DECLARE cur_employee CURSOR FOR
        SELECT Surname
        FROM Employees;
    DECLARE name CHAR(40) ;
    OPEN cur_employee;
    LOOP
        FETCH NEXT cur_employee into name ;
        .
        .
        .
    END LOOP
    CLOSE cur_employee;
END
```

Usage

One row from the result of **SELECT** is put into the variables in the variable list. The correspondence from the select list to the host variable list is one-to-one.

One or more rows from the result of **SELECT** are put either into the variables in the variable list or into the program data areas described by the named **SQLDA**. In either case, the correspondence from the select list to either the host variable list or the **SQLDA** descriptor array is one-to-one.

A cursor declared **FOR READ ONLY** sees the version of table(s) on which the cursor is declared when the cursor is opened, not the version of table(s) at the time of the first **FETCH**.

If the **FETCH** includes a positioning parameter and the position is outside the allowable cursor positions, then the **SQLLE_NOTFOUND** warning is issued.

DECLARE CURSOR must appear before **FETCH** in the C source code, and the **OPEN** statement must be executed before **FETCH**. If a host variable is being used for the cursor name, then the **DECLARE** statement actually generates code and thus must be executed before **FETCH**.

SQL Statements

In the multiuser environment, rows can be fetched by the client more than one at a time. This is referred to as block fetching or multirow fetching. The first fetch causes several rows to be sent back from the server. The client buffers these rows and subsequent fetches are retrieved from these buffers without a new request to the server.

If the `SQLSTATE_NOTFOUND` warning is returned on the fetch, then the `sqlerrd[2]` field of the `SQLCA` (`SQLCOUNT`) contains the number of rows that the attempted fetch exceeded the allowable cursor positions. (A cursor can be on a row, before the first row or after the last row.) The value is 0 if the row was not found but the position is valid, for example, executing **FETCH** with a `RELATIVE 1` clause when positioned on the last row of a cursor. The value is positive if the attempted fetch was further beyond the end of the cursor, and negative if the attempted fetch was further before the beginning of the cursor.

After successful execution of the **FETCH** statement, the `sqlerrd[1]` field of the `SQLCA` (`SQLIOCOUNT`) is incremented by the number of input/output operations required to perform the fetch. This field is actually incremented on every database statement.

The server returns in `SQLCOUNT` the number of records fetched and always returns a `SQLCOUNT` greater than zero unless there is an error. Older versions of the server only return a single row and the `SQLCOUNT` is set to zero. Thus a `SQLCOUNT` of zero with no error condition indicates one valid row has been fetched.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported in Adaptive Server Enterprise.

Permissions

The cursor must be opened and the user must have `SELECT` permission on the tables referenced in the declaration of the cursor.

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *OPEN Statement [ESQL] [SP]* on page 369
- *PREPARE Statement [ESQL]* on page 375
- *CURSOR_WINDOW_ROWS Option* on page 510

FOR Statement

Repeats the execution of a statement list once for each row in a cursor.

Syntax

```
[ statement-label: ]  
FOR for-loop-name AS cursor-name [ cursor-type ] CURSOR
```



```

    { FOR statement
  ... [ { FOR { UPDATE cursor-concurrency | FOR READ ONLY } ]
      | USING variable-name }
    DO statement-list
  END FOR [ statement-label ]

cursor-type:
  NO SCROLL
  | DYNAMIC SCROLL
  | SCROLL
  | INSENSITIVE
  | SENSITIVE

cursor-concurrency:
  BY { VALUES
  | TIMESTAMP
  | LOCK }

variable-name:
  identifier

```

Parameters

- **NO SCROLL** – a cursor declared NO SCROLL is restricted to moving forward through the result set using **FETCH NEXT** and **FETCH RELATIVE 0** seek operations. As rows cannot be returned to once the cursor leaves the row, there are no sensitivity restrictions on the cursor. When a NO SCROLL cursor is requested, the database server supplies the most efficient kind of cursor, which is an asensitive cursor.
- **DYNAMIC SCROLL** – DYNAMIC SCROLL is the default cursor type. DYNAMIC SCROLL cursors can use all formats of the **FETCH** statement. When a DYNAMIC SCROLL cursor is requested, the database server supplies an asensitive cursor. When using cursors there is always a trade-off between efficiency and consistency. Asensitive cursors provide efficient performance at the expense of consistency.
- **SCROLL** – a cursor declared SCROLL can use all formats of the **FETCH** statement. When a SCROLL cursor is requested, the database server supplies a value-sensitive cursor. The database server must execute value-sensitive cursors in such a way that result set membership is guaranteed. DYNAMIC SCROLL cursors are more efficient and should be used unless the consistent behavior of SCROLL cursors is required.
- **INSENSITIVE** – a cursor declared INSENSITIVE has its values and membership fixed over its lifetime. The result set of the **SELECT** statement is materialized when the cursor is opened. FETCHING from an INSENSITIVE cursor does not see the effect of any other **INSERT**, **UPDATE**, **MERGE**, **PUT**, or **DELETE** statement from any connection, including the connection that opened the cursor.
- **SENSITIVE** – a cursor declared SENSITIVE is sensitive to changes to membership or values of the result set.

Examples

- **Example 1** – this code fragment illustrates the use of the **FOR** loop:

```
FOR names AS curs CURSOR FOR
SELECT Surname
FROM Employees
DO
    CALL search_for_name ( Surname );
END FOR;
```

Usage

FOR is a control statement that lets you execute a list of SQL statements once for each row in a cursor.

The **FOR** statement is equivalent to a compound statement with a **DECLARE** for the cursor and a **DECLARE** of a variable for each column in the result set of the cursor, followed by a loop that fetches one row from the cursor into the local variables and executes *statement-list* once for each row in the cursor.

The name and data type of the local variables that are declared are derived from the *statement* used in the cursor. With a **SELECT** statement, the data type is the data type of the expressions in the select list. The names are the select list item aliases where they exist; otherwise, they are the names of the columns. Any select list item that is not a simple column reference must have an alias. With a **CALL** statement, the names and data types are taken from the **RESULT** clause in the procedure definition.

The **LEAVE** statement can be used to resume execution at the first statement after the **END FOR**. If the ending *statement-label* is specified, it must match the beginning *statement-label*.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported in Adaptive Server Enterprise.

Permissions

None

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *FETCH Statement [ESQL] [SP]* on page 283
- *LEAVE Statement* on page 342
- *LOOP Statement* on page 364

FOR JSON Statement

You can execute a SQL query against your database and return the results as a JSON document by using the FOR JSON clause in a SELECT statement.

JavaScript Object Notation (JSON) is a language-independent, text-based data interchange format developed for the serialization of JavaScript data. JSON also represents two structured types: objects and arrays. For more details about JSON format, see <http://www.json.org>

Syntax

```
...For JSON mode
```

```
mode :
```

```
RAW | AUTO | EXPLICIT
```

Parameters

- **mode** – controls the output format
- **RAW** – returns query results as a flattened JSON representation. Although this mode is more verbose, it can be easier to parse. The RAW clause is the recommended method for retrieving query results as JSON objects as it is the easiest method to parse and understand.
- **AUTO** – returns query results as nested JSON objects, based on query joins. Use the AUTO clause in a query when you want the result set to show the hierarchical relationship between the JSON objects.
- **EXPLICIT** – allows you to specify how column data is represented. You can specify columns as simple values, objects, or nested objects to produce uniform or heterogeneous arrays.

The EXPLICIT clause uses a column alias to provide a detailed format specification. If an alias is not present, then the given column is output as a value. An alias must be present to express a value (or object) within a nested structure. You must name the first two columns in the select-list tag and parent. A union of multiple queries can return nested JSON output by specifying the tag and parent relationship within each query.

Examples

- **Example 1** – use JSON RAW to return employee information from the Employees table:

```
SELECT
    emp.EmployeeID,
    so.CustomerID,
    so.Region
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE
emp.EmployeeID <= 195
```

SQL Statements

```
ORDER BY 1
FOR JSON RAW;
```

- **Example 2** – unlike the results returned if using FOR JSON AUTO, which would hierarchically nest the results, using FOR JSON RAW returns a flattened result set:

```
[
  { "EmployeeID" : 129, "CustomerID" : 107, "Region" :
"Eastern" },
  { "EmployeeID" : 129, "CustomerID" : 119, "Region" :
"Western" },
  ...
  { "EmployeeID" : 129, "CustomerID" : 131, "Region" :
"Eastern" },
  { "EmployeeID" : 195, "CustomerID" : 176, "Region" : "Eastern" }
]
```

- **Example 3** – unlike FOR JSON RAW, using FOR JSON AUTO returns a nested hierarchy of data, where an emp or Employee object is composed of an so or SalesOrders object that contains an array of CustomerID data:

```
[
  { "emp":
    { "EmployeeID" : 129,
      "so" : [
        { "CustomerID" : 107 , "Region" : "Eastern" },
        ...
        { "CustomerID" : 131 , "Region" : "Eastern" }
      ]
    }
  },
  { "emp" :
    { "EmployeeID" : 195,
      "so" : [
        { "CustomerID" : 109 , "Region" : "Eastern" },
        ...
        { "CustomerID" : 176 , "Region" : "Eastern" }
      ]
    }
  }
]
```

- **Example 4** – use FOR JSON EXPLICIT to return employee information from the Employees table:

```
SELECT
  1 AS tag,
  NULL AS parent,
  emp.EmployeeID AS [!!EmployeeID],
  so.CustomerID AS [!!CustomerID],
  so.Region AS [!!Region]
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE
emp.EmployeeID <= 195
ORDER BY 3
FOR JSON EXPLICIT;
```

Returns result identical to that of the FOR JSON RAW example:

```
[
  { "EmployeeID" : 129, "CustomerID" : 107, "Region" :
"Eastern" },
  { "EmployeeID" : 129, "CustomerID" : 119, "Region" :
"Western" },
  ...
  { "EmployeeID" : 129, "CustomerID" : 131, "Region" :
"Eastern" },
  { "EmployeeID" : 195, "CustomerID" : 176, "Region" : "Eastern" }
]
```

- **Example 5** – returns a result that is similar to the result of the FOR JSON AUTO example:

```
SELECT
  1          AS tag,
  NULL      AS parent,
  emp.EmployeeID AS [emp!1!EmployeeID],
  null      AS [so!2!CustomerID],
  null      AS [!2!Region]
FROM Employees as emp where emp.EmployeeID <= 195
UNION ALL
SELECT
  2,
  1,
  emp.EmployeeID,
  so.CustomerID,
  so.Region
FROM Employees as emp KEY JOIN SalesOrders as so where
emp.EmployeeID <= 195
ORDER BY 3, 1
FOR JSON EXPLICIT;
```

The above query returns the following result:

```
[
  {"emp": [{"EmployeeID":102}]},
  {"emp": [{"EmployeeID":105}]},
  {"emp":
    [{"EmployeeID":129,
      "so":[
        {"CustomerID":101,"Region":"Eastern"},
        ...
        {"CustomerID":205,"Region":"Eastern"}
      ]
    }
  ]},
  {"emp": [{"EmployeeID":148}]},
  {"emp": [{"EmployeeID":160}]},
  {"emp": [{"EmployeeID":184}]},
  {"emp": [{"EmployeeID":191}]},
  {"emp":
    [{"EmployeeID":195,
      "so":[
        {"CustomerID":101,"Region":"Eastern"},
        ...
        {"CustomerID":209,"Region":"Western"}
      ]
    }
  ]}
]
```

SQL Statements

```
    }]  
  }  
]
```

Besides the ordering of the arrays and the inclusion of employees with no sales orders, the format above differs from the FOR JSON AUTO results only in that emp is an array of structures. In FOR JSON AUTO it is understood that emp only has a single object. FOR JSON EXPLICIT uses an array encapsulation that supports aggregation.

The following example removes the emp encapsulation and returns Region as a value. This example demonstrates how the FOR JSON EXPLICIT mode provides a granular formatting control to produce something between the RAW and AUTO modes.

```
SELECT  
  1 AS tag,  
  NULL AS parent,  
  emp.EmployeeID AS [!1!EmployeeID], // remove "emp"  
encapsulation  
  null AS [so!2!id], // change  
"CustomerID" to just "id"  
  null AS [!2!] // stipulate that  
region should be emitted as a value  
FROM Employees AS emp WHERE emp.EmployeeID <= 195  
UNION ALL  
SELECT  
  2,  
  1,  
  emp.EmployeeID,  
  so.CustomerID,  
  so.Region  
FROM Employees as emp KEY JOIN SalesOrders AS so WHERE  
emp.EmployeeID <= 195  
ORDER BY 3, 1  
FOR JSON EXPLICIT;
```

In the query result, so is no longer an array of objects, but is now a two-dimensional array:

```
[  
  {"EmployeeID":102}, {"EmployeeID":105}, {"EmployeeID":129,  
  "so": [  
    [{"id":101}, "Eastern"],  
    ...  
    [{"id":205}, "Eastern"]  
  ]  
},  
  {"EmployeeID":148},  
  {"EmployeeID":160},  
  {"EmployeeID":184},  
  {"EmployeeID":191},  
  {"EmployeeID":195,  
  "so": [  
    [{"id":101}, "Eastern"],  
    ...  
    [{"id":209}, "Western"]  
  ]  
}
```

```

    ]
  }
]

```

The following example is similar to using FOR JSON RAW, but employeeID, CustomerID, and Region are output as values, not name/value pairs:

```

SELECT
    1 AS tag,
    NULL AS parent,
    emp.EmployeeID, // no alias directives
    so.CustomerID,
    so.Region
FROM Employees AS emp KEY JOIN SalesOrders AS so WHERE
emp.EmployeeID <= 195
ORDER BY 3
FOR JSON EXPLICIT;

```

The query returns the following result, where a two-dimensional array composed of EmployeeID, CustomerID, and Region is produced:

```

[
  [129,107,"Eastern"],
  ...
  [195,176,"Eastern"]
]

```

Usage

The FOR JSON clause can be used in any SELECT statement, including subqueries, queries with a GROUP BY clause or aggregate functions, and view definitions. Using the FOR JSON clause represents relational data as a JSON array composed of arrays, objects, and scalar elements.

The format for the alias directive is

```
[encapsulating_object!tag!name!qualifier]
```

where:

- **!** – delimits directive criteria.
- **encapsulating_object** – emits an encapsulating (array) object for the select-list item.
- **tag** – defines an identifier for the column used in subsequent queries. It also establishes nesting criteria (relative to its parent).
- **name** – assigns a name for the (name/value pair) object.
- **qualifier** – can be either `element` (the default), or `hide` to obfuscate the element from the result set.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

None

FORWARD TO Statement

Sends native syntax to a remote server.

Syntax

Syntax 1

```
FORWARD TO server-name { sql-statement }
```

Syntax 2

```
FORWARD TO [ server-name ]
```

Parameters

- **server-name** – the name of the remote server.
- **sql-statement** – a command in the native syntax of the remote server. The command or group of commands is enclosed in curly braces ({}) or single quotes.
- **server-name** – if you specify a *server-name*, but do not specify a statement in the **FORWARD TO** query, your session enters passthrough mode, and all subsequent queries are passed directly to the remote server. To turn passthrough mode OFF, issue the **FORWARD TO** statement without a *server_name* specification.

Note: The **FORWARD TO** statement is a server directive and cannot be used in stored procedures, triggers, events, or batches.

Examples

- **Example 1** – a passthrough session with the remote server `ase_prod`:

```
FORWARD TO aseprod
SELECT * from titles
SELECT * from authors
FORWARD TO
```

Usage

FORWARD TO enables users to specify the server to which a passthrough connection is required. The statement can be used:

- To send a statement to a remote server (Syntax 1)
- To place SAP Sybase IQ into passthrough mode for sending a series of statements to a remote server (Syntax 2)

When establishing a connection to *server-name* on behalf of the user, the server uses:

- A remote login alias set using **CREATE EXTERNLOGIN**
- If a remote login alias is not set up, the name and password used to communicate with SAP Sybase IQ

If the connection cannot be made to the server specified, the reason is contained in a message returned to the user.

After statements are passed to the requested server, any results are converted into a form that can be recognized by the client program.

Side Effects

- The remote connection is set to AUTOCOMMIT (unchained) mode for the duration of the **FORWARD TO** session. Any work that was pending prior to the **FORWARD TO** statement is automatically committed.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *CREATE EXTERNLOGIN Statement* on page 131
- *CREATE SERVER Statement* on page 192

FROM Clause

Specifies the database tables or views involved in a **SELECT** statement.

Syntax

```
...FROM table-expression [,...]

table-expression:
  table-name
  | view-name
  | procedure-name
  | common-table-expression
  | (subquery) [[ AS] derived-table-name [column_name, ...]]
  | derived-table
  | join-expression
  | ( table-expression, ... )
  | openstring-expression
```

SQL Statements

```
| apply-expression
| contains-expression
| dml-derived-table

table-name:
  [ userid. ] table-name ]
  [ [ AS ] correlation-name ]
  [ FORCE INDEX ( index-name ) ]

view-name:
  [ userid. ] view-name [ [ AS ] correlation-name ]

procedure-name:
  [ owner, ] procedure-name ([ parameter, ...])
  [ WITH ( column-name datatype, ) ]
  [ [ AS ] correlation-name ]

parameter:
  scalar-expression | table-parameter

table-parameter:
  TABLE (select-statement) [ OVER ( table-parameter-over ) ]

table-parameter-over:
  [ PARTITION BY { ANY
  | NONE | table-expression } ]
  [ ORDER BY { expression | integer }
  [ ASC | DESC ] [, ...] ]

derived-table:
  ( select-statement )
  [ AS ] correlation-name [ ( column-name, ... ) ]

join-expression:
  table-expression join-operator table-expression
  [ ON join-condition ]

join-operator:
  [ KEY | NATURAL ] [ join-type ] JOIN | CROSS JOIN

join-type:
  INNER
  | LEFT [ OUTER ]
  | RIGHT [ OUTER ]
  | FULL [ OUTER ]

openstring-expression:
  OPENSTRING ( { FILE | VALUE } string-expression )
  WITH ( rowset-schema )
  [ OPTION ( scan-option ... ) ]
  [ AS ] correlation-name

apply-expression:
  table-expression { CROSS | OUTER } APPLY table-expression

contains-expression:
```

```

{ table-name | view-name } CONTAINS
( column-name [...], contains-query )
[ [ AS ] score-correlation-name ]

rowset-schema:
  column-schema-list
    | TABLE [owner.]table-name [ ( column-list ) ]

column-schema-list:
  { column-name user-or-base-type | filler( ) } [ , ... ]

column-list:
  { column-name | filler( ) } [ , ... ]

scan-option:
  BYTE ORDER MARK { ON | OFF }
  | COMMENTS INTRODUCED BY comment-prefix
  | DELIMITED BY string
  | ENCODING encoding
  | ESCAPE CHARACTER character
  | ESCAPES { ON | OFF }
  | FORMAT { TEXT | BCP }
  | HEXADECIMAL { ON | OFF }
  | QUOTE string
  | QUOTES { ON | OFF }
  | ROW DELIMITED BY string
  | SKIP integer
  | STRIP { ON | OFF | LTRIM | RTRIM | BOTH }

contains-query:
  string

dml-derived-table:
  ( dml-statement ) REFERENCING ( [ table-version-names | NONE ] )

dml-statement:
  insert-statement
  update-statement
  delete-statement

table-version-names:
  OLD [ AS ] correlation-name [ FINAL [ AS ] correlation-name ]
  | FINAL [ AS ] correlation-name

```

Parameters

- **table-name** – a base table or temporary table. Tables owned by a different user can be qualified by specifying the user ID. Tables owned by groups to which the current user belongs are found by default without specifying the user ID.
- **view-name** – specifies a view to include in the query. As with tables, views owned by a different user can be qualified by specifying the user ID. Views owned by groups to which

the current user belongs are found by default without specifying the user ID. Although the syntax permits table hints on views, these hints have no effect.

- **procedure-name** – a stored procedure that returns a result set. This clause applies to the FROM clause of SELECT statements only. The parentheses following the procedure name are required even if the procedure does not take parameters. DEFAULT can be specified in place of an optional parameter.
- **parameter** – specifies a scalar-parameter or table-parameter clause. A scalar-parameter are any objects of a valid SQL datatype. A table-parameter can be specified using a table, view or common table-expression name which are treated as new instance of this object if the object is also used outside the table-parameter.

This query illustrates a valid **FROM** clause where the two references to the same table T are treated as two different instances of the same table T.

```
SELECT * FROM T, my_proc(TABLE(SELECT T.Z, T.X FROM T)
OVER(PARTITION BY T.Z));
```

Table Parameterized Function (TPF) Example—This query illustrates a valid **FROM** clause.

```
SELECT * FROM R, SELECT * FROM my_udf(1);
SELECT * FROM my_tpf(1, TABLE(SELECT c1, c2 FROM t)
(my_proc(R.X, TABLE T OVER PARTITION BY T.X)) AS XX;
```

If a subquery is used to define the TABLE parameter, then the following restrictions must hold:

- The table-parameter clause must be of type IN.
- PARTITION BY or ORDER BY clauses must refer to the columns of the derived table and outer references. An expression in the expression-list can be an integer K which refers to the Kth column of the TABLE input parameter.

Note: A Table UDF can only be referenced in a **FROM** clause of a SQL statement.

- **PARTITION BY** – logically specifies how the invocation of the function will be performed by the execution engine. The execution engine must invoke the function for each partition and the function must process a whole partition in each invocation.

PARTITION BY clause also specifies how the input data must be partitioned such that each invocation of the function will process exactly one partition of data. The function must be invoked the number of times equal to the number of partitions. For TPF, the parallelism characteristics are established through dynamic negotiation between the server and the UDF at the runtime. If the TPF can be executed in parallel, for N input partitions, the function can be instantiated M times, with $M \leq N$. Each instantiation of the function can be invoked more than once, each invocation consuming exactly one partition.

You can specify only one TABLE input parameter for PARTITION BY *expression-list* or PARTITION BY ANY clause. For all other TABLE input parameters you must specify, explicit or implicit PARTITION BY NONE clause.

Note: The execution engine can invoke the function in any order of the partitions and the function is assumed to return the same result sets regardless of the partitions order. Partitions cannot be split among two invocations of the function.

- **ORDER BY** – specifies that the input data in each partition is expected to be sorted by *expression-list* by the execution engine. The UDF expects each partition to have this physical property. If only one partition exists, the whole input data is ordered based on the ORDER BY specification. ORDER BY clause can be specified for any of the TABLE input parameters with PARTITION BY NONE or without PARTITION BY clause.
- **derived-table** – you can supply a SELECT statement instead of table or view name in the FROM clause. A SELECT statement used in this way is called a derived table, and it must be given an alias. For example, the following statement contains a derived table, MyDerivedTable, which ranks products in the Products table by UnitPrice.

```
SELECT TOP 3 *
  FROM ( SELECT Description,
             Quantity,
             UnitPrice,
             RANK() OVER ( ORDER BY UnitPrice ASC )
             AS Rank
        FROM Products ) AS MyDerivedTable
 ORDER BY Rank;
```

- **join-expression, join-operator, join-type** – the join-type keywords are:

Keyword	Description
CROSS JOIN	Returns the Cartesian product (cross product) of the two source tables
NATURAL JOIN	Compares for equality all corresponding columns with the same names in two tables (a special case equijoin; columns are of same length and data type)
KEY JOIN	Restricts foreign-key values in the first table to be equal to the primary-key values in the second table
INNER JOIN	Discards all rows from the result table that do not have corresponding rows in both tables
LEFT OUTER JOIN	Preserves unmatched rows from the left table, but discards unmatched rows from the right table
RIGHT OUTER JOIN	Preserves unmatched rows from the right table, but discards unmatched rows from the left table
FULL OUTER JOIN	Retains unmatched rows from both the left and the right tables

Do not mix comma-style joins and keyword-style joins in the **FROM** clause. The same query can be written two ways, each using one of the join styles. The ANSI syntax keyword style join is preferable.

This query uses a comma-style join:

```
SELECT *
  FROM Products pr, SalesOrders so, SalesOrderItems si
 WHERE pr.ProductID = so.ProductID
       AND pr.ProductID = si.ProductID;
```

The same query can use the preferable keyword-style join:

```
SELECT *
  FROM Products pr INNER JOIN SalesOrders so
       ON (pr.ProductID = so.ProductID)
     INNER JOIN SalesOrderItems si
       ON (pr.ProductID = si.ProductID);
```

The ON clause filters the data of inner, left, right, and full joins. Cross joins do not have an ON clause. In an inner join, the ON clause is equivalent to a WHERE clause. In outer joins, however, the ON and WHERE clauses are different. The ON clause in an outer join filters the rows of a cross product and then includes in the result the unmatched rows extended with nulls. The WHERE clause then eliminates rows from both the matched and unmatched rows produced by the outer join. You must take care to ensure that unmatched rows you want are not eliminated by the predicates in the WHERE clause.

You cannot use subqueries inside an outer join ON clause.

- **openstring-expression** – Specify an OPENSTRING clause to query within a file or a BLOB, treating the content of these sources as a set of rows. When doing so, you also specify information about the schema of the file or BLOB for the result set to be generated, since you are not querying a defined structure such as a table or view. This clause applies to the FROM clause of a SELECT statement. It is not supported for UPDATE or DELETE statements.
- **apply-expression** – Use this clause to specify a join condition where the right table-expression is evaluated for every row in the left table-expression. For example, you can use an apply expression to evaluate a function, procedure, or derived table for each row in a table expression.
- **contains-expression** – Use the CONTAINS clause after a table name to filter the table, and return only those rows matching the full text query specified with contains-query. Every matching row of the table is returned, along with a score column that can be referred to using score-correlation-name, if it is specified. If score-correlation-name is not specified, then the score column can be referred to by the default correlation name, contains.
- **dml-derived-table** – Supports the use of a DML statement (INSERT, UPDATE, or DELETE) as a table expression in a query's FROM clause.

Examples

- **Example 1** – these are valid FROM clauses:

```
...
FROM Employees
```

```

...
...
FROM Employees NATURAL JOIN Departments
...
...
FROM Customers
KEY JOIN SalesOrders
KEY JOIN SalesOrderItems
KEY JOIN Products
...

```

- **Example 2** – this query illustrates how to use derived tables in a query:

```

SELECT Surname, GivenName, number_of_orders
FROM Customers JOIN
    ( SELECT CustomerID, count(*)
      FROM SalesOrders
      GROUP BY CustomerID )
  AS sales_order_counts ( CustomerID,
                          number_of_orders )
ON ( Customers.ID = sales_order_counts.cust_id )
WHERE number_of_orders > 3

```

Usage

The **SELECT** statement requires a table list to specify which tables are used by the statement.

Note: Although this description refers to tables, it also applies to views unless otherwise noted.

The **FROM** table list creates a result set consisting of all the columns from all the tables specified. Initially, all combinations of rows in the component tables are in the result set, and the number of combinations is usually reduced by join conditions and/or **WHERE** conditions.

Tables owned by a different user can be qualified by specifying the *userid*. Tables owned by roles to which the current user belongs are found by default without specifying the user ID.

The correlation name is used to give a temporary name to the table for this SQL statement only. This is useful when referencing columns that must be qualified by a table name but the table name is long and cumbersome to type. The correlation name is also necessary to distinguish between table instances when referencing the same table more than once in the same query. If no correlation name is specified, then the table name is used as the correlation name for the current statement.

If the same correlation name is used twice for the same table in a table expression, that table is treated as if it were only listed once. For example, in:

```

SELECT *
FROM SalesOrders
KEY JOIN SalesOrderItems,
SalesOrders
KEY JOIN Employees

```

The two instances of the `SalesOrders` table are treated as one instance that is equivalent to:

SQL Statements

```
SELECT *  
FROM SalesOrderItems  
KEY JOIN SalesOrders  
KEY JOIN Employees
```

By contrast, the following is treated as two instances of the `Person` table, with different correlation names `HUSBAND` and `WIFE`.

```
SELECT *  
FROM Person HUSBAND, Person WIFE
```

Join columns require like data types for optimal performance.

- **Performance Considerations** – Depending on the query, SAP Sybase IQ allows between 16 and 64 tables in the **FROM** clause with the optimizer turned on; however, performance might suffer if you have more than 16 to 18 tables in the **FROM** clause in very complex queries.

Note: If you omit the **FROM** clause, or if all tables in the query are in the `SYSTEM` dbspace, the query is processed by SQL Anywhere instead of SAP Sybase IQ and might behave differently, especially with respect to syntactic and semantic restrictions and the effects of option settings.

If you have a query that does not require a **FROM** clause, you can force the query to be processed by SAP Sybase IQ by adding the clause **FROM iq_dummy**, where `iq_dummy` is a one-row, one-column table that you create in your database.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—The **JOIN** clause is not supported in some versions of Adaptive Server Enterprise. Instead, you must use the **WHERE** clause to build joins.

Permissions

Must be connected to the database.

See also

- *DELETE Statement* on page 248
- *SELECT Statement* on page 417

GET DESCRIPTOR Statement [ESQL]

Retrieves information about variables within a descriptor area, or retrieves actual data from a variable in a descriptor area.

Syntax

```
GET DESCRIPTOR descriptor-name
  { ...hostvar = COUNT } | VALUE n assignment [,...] }
```

assignment:

```
hostvar = { TYPE
  | LENGTH
  | PRECISION
  | SCALE
  | DATA
  | INDICATOR
  | NAME
  | NULLABLE
  | RETURNED_LENGTH }
```

Examples

- **Example 1** – for an example, see *ALLOCATE DESCRIPTOR Statement [ESQL]*.

Usage

The value *n* specifies the variable in the descriptor area about which information is retrieved.

Type checking is performed when doing **GET DESCRIPTOR ... DATA** to ensure that the host variable and the descriptor variable have the same data type. LONG VARCHAR and LONG BINARY are not supported by **GET DESCRIPTOR ... DATA**.

If an error occurs, it is returned in the SQLCA.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *ALLOCATE DESCRIPTOR Statement [ESQL]* on page 5
- *DEALLOCATE DESCRIPTOR Statement [ESQL]* on page 237

- *SET DESCRIPTOR Statement [ESQL]* on page 430

GOTO Statement [T-SQL]

Branches to a labeled statement.

Syntax

```
label : GOTO label
```

Examples

- **Example 1** – this Transact-SQL batch prints the message “yes” on the server window four times:

```
declare @count smallint
select @count = 1
restart:
    print 'yes'
    select @count = @count + 1
    while @count <=4
        goto restart
```

Usage

Any statement in a Transact-SQL procedure or batch can be labeled. The label name is a valid identifier followed by a colon. In the **GOTO** statement, the colon is not used.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Adaptive Server Enterprise supports the GOTO statement.

Permissions

None

GRANT CHANGE PASSWORD Statement

Allows users to manage passwords for other users and administer the CHANGE PASSWORD system privilege.

Syntax

```
GRANT CHANGE PASSWORD ( target_user_list | ANY | ANY WITH ROLES
target_role_list )
    TO userID [,...]
    [ WITH ADMIN [ONLY] OPTION | WITH NO ADMIN OPTION]
```

Parameters

- **target_user_list** – users the grantee has the potential to impersonate. The list must consist of existing users or user-extended roles with login passwords. Separate the userIDs in the list with commas.
- **ANY** – all database users with login passwords become potential target users to manage passwords for each grantee.
- **ANY WITH ROLES target_role_list** – list of target roles for each grantee. Any users who are granted any of the target roles become potential target users for each grantee. The *target_role_list* must consist of existing roles and the users who are granted said roles must consist of database users with login passwords. Use commas to separate multiple userIDs.
- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
- **WITH ADMIN OPTION** – (valid with the ANY clause only) The user can both manage passwords and grant the CHANGE PASSWORD system privilege to another user.
- **WITH ADMIN ONLY OPTION** – (valid with the ANY clause only) The user can grant the CHANGE PASSWORD system privilege to another user, but cannot manage passwords of other users.
- **WITH NO ADMIN OPTION** – the user can manage passwords, but cannot grant the CHANGE PASSWORD system privilege to another user.

Examples

- **Example 1** – grants Sally and Laurel the ability to manage the password of Bob, Sam, and Peter:

```
GRANT CHANGE PASSWORD (Bob, Sam, Peter) TO (Sally, Laurel)
```

- **Example 2** – grants Mary the right to grant the CHANGE PASSWORD system privilege to any user in the database. However, since the system privilege is granted with the WITH ADMIN ONLY OPTION clause, Mary cannot manage the password of any other user.

```
GRANT CHANGE PASSWORD (ANY) TO Mary WITH ADMIN ONLY OPTION
```

- **Example 3** – grants Steve and Joe the ability to manage the password of any member of Role1 or Role2:

```
GRANT CHANGE PASSWORD (ANY WITH ROLES Role1, Role2) TO Steve, Joe
```

Usage

A user can be granted the ability to manage the password of any user in the database (ANY) or only specific users (*target_users_list*) or members of specific roles (ANY WITH ROLES *target_roles_list*). Administrative rights to the CHANGE PASSWORD system privilege can only be granted when using the ANY clause.

SQL Statements

If no clause is specified, ANY is used by default. If no administrative clause is specified in the grant statement, the WITH NO ADMIN OPTION clause is used.

By default, the CHANGE PASSWORD system privilege is granted to the SYS_AUTH_SA_ROLE compatibility role with the WITH NO ADMIN OPTION clause and to the SYS_AUTH_SSO_ROLE compatibility role with the ADMIN ONLY OPTION clause, if they exist.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

- Requires the CHANGE PASSWORD system privilege granted with administrative rights.
- Each target user specified (target_users_list) is an existing user or user-extended role with a login password.
- Each target role specified (target_roles_list) must be an existing user-extended or user-defined role.

See also

- *ALTER USER Statement* on page 71
- *REVOKE CHANGE PASSWORD Statement* on page 398

GRANT CONNECT Statement

Grants CONNECT privilege to a user.

Syntax

```
GRANT CONNECT
  TO userID [,...]
  IDENTIFIED BY password [,...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples

- **Example 1** – creates two new users for the database named Laurel and Hardy:

```
GRANT CONNECT TO Laurel, Hardy
IDENTIFIED BY Stan, Ollie
```

- **Example 2** – creates user Jane with no password:

```
GRANT CONNECT TO Jane
```

- **Example 3** – changes the password for Bob to newpassword:

```
GRANT CONNECT TO Bob IDENTIFIED BY newpassword
```

Usage

GRANT CONNECT can be used to create a new user or also be used by any user to change their own password.

Tip: Use the **CREATE USER** statement rather than the **GRANT CONNECT** statement to create users.

If you inadvertently enter the user ID of an existing user when you are trying to add a new user, you are actually changing the password of the existing user. You do not receive a warning because this behavior is considered normal.

The stored procedures **sp_addlogin** and **sp_adduser** can also be used to add users. These procedures display an error if you try to add an existing user ID.

Note: Use system procedures, not **GRANT** and **REVOKE** statements to add and remove user IDs.

A user without a password cannot connect to the database. This is useful when you are creating groups and you do not want anyone to connect to the role user ID. To create a user without a password, do not include the **IDENTIFIED BY** clause.

When specifying a password, it must be a valid identifier. Passwords have a maximum length of 255 bytes. If the **VERIFY_PASSWORD_FUNCTION** database option is set to a value other than the empty string, the **GRANT CONNECT TO** statement calls the function identified by the option value. The function returns **NULL** to indicate that the password conforms to rules. If the **VERIFY_PASSWORD_FUNCTION** option is set, you can specify only one *userid* and *password* with the **GRANT CONNECT** statement.

Invalid names for database user IDs and passwords include those that:

- Begin with white space or single or double quotes
- End with white space
- Contain semicolons

Standards

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – The security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

- If you are creating a new user, you must have the **MANAGE ANY USER** system privilege.
- Any user can change his or her own password.
- If you are changing another user's password, you must have the **CHANGE PASSWORD** system privilege.

Note: If you are changing another user's password, the other user cannot be connected to the database.

See also

- *CREATE USER Statement* on page 230
- *REVOKE CONNECT Statement* on page 399

GRANT CREATE Statement

Grants **CREATE** privilege on a specified dbspace to the specified users and roles.

Syntax

```
GRANT CREATE
ON dbspace_name
TO userID [,...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples

- **Example 1** – grants users Lawrence and Swift **CREATE** privilege on dbspace *DspHist*:

```
GRANT CREATE ON DspHist
TO LAWRENCE, SWIFT
```

- **Example 2** – grants **CREATE** privilege on dbspace *DspHist* to users Fiona and Ciaran:

```
GRANT CREATE ON DspHist TO Fiona, Ciaran
```

Standards

- **SQL** – other syntaxes are vendor extensions to ISO/ANSI SQL grammar.

- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires the `MANAGE ANY DBSPACE` system privilege.

See also

- *REVOKE CREATE Statement* on page 400

GRANT Object-Level Privilege Statement

Grants database object-level privileges on individual tables or views to a user or role.

Syntax

```
GRANT object-level-privilege [, ...]
  ON [ owner.]object-name
  TO userID [,...]
  [ WITH GRANT OPTION ]

object-level-privilege:
  ALL [ PRIVILEGES ]
  | ALTER
  | DELETE
  | INSERT
  | LOAD
  | REFERENCE [ ( column-name [, ...] ) ]
  | SELECT [ ( column-name [, ...] ) ]
  | TRUNCATE
  | UPDATE [ ( column-name, ... ) ] }
```

Parameters

- **userID** – must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate the userIDs in the list with commas.
- **ALL** – grants all privileges to users
- **ALTER** – users can alter this table with the **ALTER TABLE** statement. This privilege is not allowed for views.
- **DELETE** – users can delete rows from this table or view.
- **INSERT** – users can insert rows into the named table or view.
- **LOAD** – users can load data into the named table or view.
- **REFERENCES** – users can create indexes on the named tables, and foreign keys that reference the named tables. If column names are specified, then users can reference only

SQL Statements

those columns. REFERENCES privileges on columns cannot be granted for views, only for tables.

- **SELECT** – users can look at information in this view or table. If column names are specified, then the users can look at only those columns. SELECT permissions on columns cannot be granted for views, only for tables.
- **TRUNCATE** – users can truncate the named table or view.
- **UPDATE** – users can update rows in this view or table. If column names are specified, users can update only those columns. UPDATE privileges on columns cannot be granted for views, only for tables. To update a table, users must have both SELECT and UPDATE privilege on the table.
- **WITH GRANT OPTION** – the named user ID is also given privileges to grant the same privileges to other user IDs.

Usage

You can list the table privileges, or specify ALL to grant all privileges at once.

Standards

- SQL – Syntax is an entry-level feature.
- Sybase – Syntax is supported in Adaptive Server Enterprise.

Permissions

Requires one of:

- MANAGE ANY OBJECT PRIVILEGE system privilege
- You have been granted the specific object privilege with the WITH GRANT OPTION clause on the table.
- You own of the table.

See also

- *REVOKE Object-Level Privilege Statement* on page 403

GRANT EXECUTE Statement

Grants EXECUTE privilege on a procedure or user-defined function.

Syntax

```
GRANT EXECUTE
ON [ owner. ] { procedure-name | user-defined-function-name }
TO userID [ , ... ]
```


Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL – syntax is a Persistent Stored Module feature.
- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires one of:

- **MANAGE ANY OBJECT PRIVILEGE** system privilege.
- You own the procedure.

See also

- *REVOKE EXECUTE Statement* on page 401

GRANT INTEGRATED LOGIN Statement

Creates an explicit integrated login mapping between one or more Windows user profiles and an existing database user ID. This allows a user who successfully logged in to their local machine to connect to a database without having to provide a user ID or password.

Syntax

```
GRANT INTEGRATED LOGIN
  TO user_profile_name [, ...]
  AS USER userID [, ...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL – other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *REVOKE INTEGRATED LOGIN Statement* on page 401

GRANT KERBEROS LOGIN Statement

Creates a Kerberos-authenticated login mapping from one or more Kerberos principals to an existing database user ID. This allows a user who has successfully logged in to Kerberos (user who has a valid Kerberos ticket-granting ticket) to connect to a database without having to provide a user ID or password.

Syntax

```
GRANT KERBEROS LOGIN  
TO client-Kerberos-principal [, ...]  
AS USER userID [,...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL – other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *REVOKE KERBEROS LOGIN Statement* on page 402

GRANT ROLE Statement

Grants roles to users or other roles, with or without administrative rights.

Syntax

```
GRANT ROLE role_name [, ...]  
TO grantee [, ...]
```

```

[ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
[ WITH NO SYSTEM PRIVILEGE INHERITANCE ]

role_name:
  dbo†††
  | diagnostics†††
  | PUBLIC†††
  | rs_systabgroup†††
  | SA_DEBUG†††
  | SYS†††
  | SYS_AUTH_SA_ROLE
  | SYS_AUTH_SSO_ROLE
  | SYS_AUTH_DBA_ROLE††
  | SYS_AUTH_RESOURCE_ROLE†
  | SYS_AUTH_BACKUP_ROLE†
  | SYS_AUTH_VALIDATE_ROLE†
  | SYS_AUTH_WRITEFILE_ROLE
  | SYS_AUTH_WRITEFILECLIENT_ROLE
  | SYS_AUTH_READFILE_ROLE
  | SYS_AUTH_READFILECLIENT_ROLE
  | SYS_AUTH_PROFILE_ROLE
  | SYS_AUTH_USER_ADMIN_ROLE
  | SYS_AUTH_SPACE_ADMIN_ROLE
  | SYS_AUTH_MULTIPLEX_ADMIN_ROLE
  | SYS_AUTH_OPERATOR_ROLE
  | SYS_AUTH_PERMS_ADMIN_ROLE
  | SYS_REPLICATE_ADMIN_ROLE†††
  | SYS_RUN_REPLICATE_ROLE†††
  | SYS_SPATIAL_ADMIN_ROLE†††
  | user-defined role name

```

- The WITH NO SYSTEM PRIVILEGE INHERITANCE clause can be used when granting select compatibility roles to other roles. It prevents automatic inheritance of the compatibility role's underlying system privileges by members of the role. When granted to user-extended roles, the WITH NO SYSTEM PRIVILEGE INHERITANCE clause applies to members of the role only. The user acting as a role automatically inherits the underlying system privileges regardless of the clause.
- The WITH NO ADMIN OPTION WITH NO SYSTEM PRIVILEGE INHERITANCE and WITH NO SYSTEM PRIVILEGE INHERITANCE clauses are semantically equivalent.
- [†]The WITH ADMIN OPTION or WITH ADMIN ONLY clauses can not be specified in combination with the WITH NO SYSTEM PRIVILEGE INHERITANCE clause when granting the SYS_AUTH_BACKUP_ROLE, SYS_AUTH_RESOURCE_ROLE, or SYS_AUTH_VALIDATE_ROLE roles.
- ^{††}The WITH ADMIN OPTION clause can only be specified in combination with the WITH NO SYSTEM PRIVILEGE INHERITANCE clause when granting the SYS_AUTH_DBA_ROLE or SYS_RUN_REPLICATION_ROLE roles.
- ^{†††}The WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are not supported for system roles.

Parameters

- **role_name** – must already exist in the database. Separate multiple role names with commas.
- **grantee** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
- **WITH NO ADMIN OPTION** – each *grantee* is granted the underlying system privileges of each *role_name*, but cannot grant *role_name* to another user.
- **WITH ADMIN ONLY OPTION** – each *userID* is granted administrative privileges over each *role_name*, but not the underlying system privileges of *role_name*.
- **WITH ADMIN OPTION** – each *userID* is granted the underlying system privileges of each *role_name*, along with the ability to grant *role_name* to another user.
- **WITH NO SYSTEM PRIVILEGE INHERITANCE** – the underlying system privileges of the granting role are not inherited by the members of the receiving role. However, if the receiving role is a user-extended role, the underlying system privileges are granted to the extended user.

Examples

- **Example 1** – grants `Sales_Role` to `Sally`, with administrative privileges, which means she can grant or revoke `Sales_Role` to other users as well as perform any authorized tasks granted by the role:

```
GRANT ROLE Sales_Role TO Sally WITH ADMIN OPTION
```

- **Example 2** – grants the compatibility role `SYS_AUTH_PROFILE_ROLE` to the role `Sales_Admin` with no administrative rights. `Sales_Admin` is a standalone role and `Mary` and `Peter` have been granted `Sales_Admin`. Since `SYS_AUTH_PROFILE_ROLE` is an inheritable compatibility role, `Mary` and `Peter` are granted the underlying system privileges of `Sales_Role`. Since the role is granted with no administrative rights, they cannot grant or revoke the role.

```
GRANT ROLE SYS_AUTH_PROFILE_ROLE TO Sales_Role WITH NO ADMIN  
OPTION
```

- **Example 3** – grants the compatibility role `SYS_AUTH_BACKUP_ROLE` to `Tom` with no administrative rights. `Tom` is a user-extended role to which `Betty` and `Laurel` have been granted. Since `SYS_AUTH_BACKUP_ROLE` is a non-inheritable compatibility role, the underlying system privileges of the role are not granted to `Betty` and `Laurel`. However, since `Tom` is an extended user, the underlying system privileges are granted directly to `Tom`.

```
GRANT ROLE SYS_AUTH_BACKUP_ROLE TO Tom  
WITH NO SYSTEM PRIVILEGE INHERITANCE
```

Usage

Use of the WITH ADMIN OPTION or WITH ADMIN ONLY OPTION clause allows the grantee to grant or revoke the role, but does not allow the grantee to drop the role.

By default, if no administrative clause is specified in the grant statement, each compatibility role is granted with these default administrative rights:

WITH ADMIN OPTION	WITH ADMIN ONLY OPTION	WITH NO ADMIN OPTION
SYS_AUTH_SA_ROLE SYS_AUTH_SSO_ROLE	SYS_AUTH_DBA_ROLE	SYS_AUTH_RE- SOURCE_ROLE SYS_AUTH_BACK- UP_ROLE SYS_AUTH_VALI- DATE_ROLE SYS_AUTH_WRITE- FILE_ROLE SYS_AUTH_WRITEFILE- CLIENT_ROLE SYS_AUTH_READ- FILE_ROLE SYS_AUTH_READFILE- CLIENT_ROLE SYS_AUTH_PRO- FILE_ROLE SYS_AUTH_USER_AD- MIN_ROLE SYS_AUTH_SPACE_AD- MIN_ROLE SYS_AUTH_MULTI- PLEX_ADMIN_ROLE SYS_AUTH_OPERA- TOR_ROLE SA_DEBUG SYS_RUN_REPLICA- TION_ROLE

The SYS_AUTH_PERMS_ADMIN_ROLE role grants these underlying roles with these default administrative rights:

WITH ADMIN OPTION	WITH NO ADMIN OPTION
SYS_AUTH_BACKUP_ROLE	MANAGE ROLES
SYS_AUTH_OPERATOR_ROLE	MANAGE ANY OBJECT PRIVILEGE
SYS_AUTH_USER_ADMIN_ROLE	CHANGE PASSWORD
SYS_AUTH_SPACE_ADMIN_ROLE	
SYS_AUTH_MULTIPLEX_ADMIN_ROLE	
SYS_AUTH_RESOURCE_ROLE	
SYS_AUTH_VALIDATE_ROLE	
SYS_AUTH_PROFILE_ROLE	
SYS_AUTH_WRITEFILE_ROLE	
SYS_AUTH_WRITEFILECLIENT_ROLE	
SYS_AUTH_READFILE_ROLE	
SYS_AUTH_READFILECLIENT_ROLE	

Standards

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – Syntax is supported in Adaptive Server Enterprise.

Permissions

- Requires MANAGE ROLES system privilege to grant these system roles:
 - dbo
 - diagnostics
 - PUBLIC
 - rs_systabgroup
 - SA_DEBUG SYS
 - SYS
 - SYS_REPLICATION_ADMIN_ROLE
 - SYS_RUN_REPLICATION_ROLE
 - SYS_SPATIAL_ADMIN_ROLE
- Requires administrative privilege over the role to grant these roles:
 - SYS_AUTH_SA_ROLE
 - SYS_AUTH_SSO_ROLE
 - SYS_AUTH_DBA_ROLE
 - SYS_AUTH_RESOURCE_ROLE
 - SYS_AUTH_BACKUP_ROLE

- SYS_AUTH_VALIDATE_ROLE
- SYS_AUTH_WRITEFILE_ROLE
- SYS_AUTH_WRITEFILECLIENT_ROLE
- SYS_AUTH_READFILE_ROLE
- SYS_AUTH_READFILECLIENT_ROLE
- SYS_AUTH_PROFILE_ROLE
- SYS_AUTH_USER_ADMIN_ROLE
- SYS_AUTH_SPACE_ADMIN_ROLE
- SYS_AUTH_MULTIPLEX_ADMIN_ROLE
- SYS_AUTH_OPERATOR_ROLE
- SYS_AUTH_PERMS_ADMIN_ROLE
- <user-defined role name>

See also

- *CREATE USER Statement* on page 230
- *REVOKE System Privilege Statement* on page 408
- *REVOKE ROLE Statement* on page 404
- *VERIFY_PASSWORD_FUNCTION Option* on page 659

GRANT SET USER Statement

Grants the ability for one user to impersonate another user and to administer the SET USER system privilege.

Syntax

```
GRANT SET USER ( target_users_list
                | ANY
                | ANY WITH ROLES target_roles_list )
TO userID [,...]
[ WITH ADMIN [ ONLY ] OPTION | WITH NO ADMIN OPTION ]
```

Parameters

- **target_users_list** – must consist of existing users with login passwords and is the potential list of target users who can no longer be impersonated by grantee users. Separate the user IDs in the list with commas.
- **ANY** – the potential list of target users for each grantee consists of all database users with login passwords.
- **ANY WITH ROLES target_roles_list** – the *target_role_list* must consist of existing roles, and the potential list of target users for each grantee must consist of database users

SQL Statements

with login passwords that have a subset of roles in *target_role_list*. Separate the list of roles with commas.

- **userID** – each *userID* must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate the userIDs in the list with commas.
- **WITH ADMIN OPTION** – (valid in conjunction with the ANY clause only) The user can both issue the SETUSER command to impersonate another user and grant the SET USER system privilege to another user.
- **WITH ADMIN ONLY OPTION** – (valid in conjunction with the ANY clause only) The user can grant the SET USER system privilege to another user, but cannot issue the SETUSER command to impersonate another user.
- **WITH NO ADMIN OPTION** – the user can issue the SETUSER command to impersonate another user, but cannot grant the SET USER system privilege to another user.

Examples

- **Example 1** – grants Sally and Laurel the ability to impersonate Bob, Sam, and Peter:

```
GRANT SET USER (Bob, Sam, Peter) TO (Sally, Laurel)
```

- **Example 2** – grants Mary the right to grant the SET USER system privilege to any user in the database. However, since the system privilege is granted with the WITH ADMIN ONLY OPTION clause, Mary cannot impersonate any other user.

```
GRANT SET USER (ANY) TO Mary WITH ADMIN ONLY OPTION
```

- **Example 3** – grants Steve and Joe the ability to impersonate any member of Role1 or Role2:

```
GRANT SET USER (ANY WITH ROLES Role1, Role2) TO Steve, Joe
```

Usage

A user can be granted the ability to impersonate any user in the database (ANY) or only specific users (*target_users_list*) or members of specific roles (ANY WITH ROLES *target_roles_list*). Administrative rights to the SET USER system privilege can only be granted when using the ANY clause.

If no clause is specified, ANY is used by default. If no administrative clause is specified in the grant statement, the WITH NO ADMIN OPTION clause is used.

If regranteeing the SET USER system privilege to a user, the effect of the regrant is cumulative.

By default, the SET USER system privilege is granted to the SYS_AUTH_SSO_ROLE compatibility role with the WITH NO ADMIN OPTION clause, if they exist.

The granting of the SET USER system privilege to a user only grants the potential to impersonate another user. Validation of the *at-least* criteria required to successfully impersonate another user does not occur until the **SETUSER** statement is issued.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

- Requires the SET USER system privilege granted with administrative rights.
- Each target user specified (target_users_list) is an existing user or user-extended role with a login password.
- Each target role specified (target_roles_list) must be an existing user-extended or user-defined role.

See also

- *REVOKE SET USER Statement* on page 407

GRANT System Privilege Statement

Grants specific system privileges to users or roles, with or without administrative rights.

Syntax

```
GRANT system_privilege_name [, ...]
  TO userID [, ...]
  [ { WITH NO ADMIN | WITH ADMIN [ ONLY ] } OPTION ]
```

Parameters

- **system_privilege** – must be the name of an existing system privilege.
- **userID** – must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate multiple userIDs with commas.
- **WITH NO ADMIN OPTION** – the user can manage the system privilege, but cannot grant the system privilege to another user.
- **WITH ADMIN ONLY OPTION** – If the WITH ADMIN ONLY OPTION clause is used, each *userID* is granted administrative privileges over each *system_privilege*, but NOT the *system_privilege* itself.
- **WITH ADMIN OPTION** – each *userID* is granted administrative privileges over each *system_privilege* in addition to all underlying system privileges of *system_privilege*.

Examples

- **Example 1** – grants the DROP CONNECTION system privilege to Joe with administrative privileges:

```
GRANT DROP CONNECTION TO Joe WITH ADMIN OPTION
```

- **Example 2** – grants the CHECKPOINT system privilege to Sally with no administrative privileges:

```
GRANT CHECKPOINT TO Sally WITH NO ADMIN OPTION
```

- **Example 3** – grants the MONITOR system privilege to Jane with administrative privileges only:

```
GRANT MONITOR TO Jane WITH ADMIN ONLY OPTION
```

Usage

By default, if no administrative clause is specified in the grant statement, the WITH NO ADMIN OPTION clause is used.

Standards

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – Syntax is supported in Adaptive Server Enterprise.

Permissions

Requires administrative privilege over the system privilege being granted.

See also

- *CREATE USER Statement* on page 230
- *REVOKE System Privilege Statement* on page 408
- *REVOKE ROLE Statement* on page 404
- *VERIFY_PASSWORD_FUNCTION Option* on page 659

List of All System Privileges

A list of all system privileges.

System privileges control the rights of users to perform authorized database tasks.

The following is a list of available system privileges:

- ACCESS SERVER LS system privilege
- ALTER ANY INDEX system privilege
- ALTER ANY MATERIALIZED VIEW system privilege
- ALTER ANY OBJECT system privilege

- ALTER ANY OBJECT OWNER system privilege
- ALTER ANY PROCEDURE system privilege
- ALTER ANY SEQUENCE system privilege
- ALTER ANY TABLE system privilege
- ALTER ANY TEXT CONFIGURATION system privilege
- ALTER ANY TRIGGER system privilege
- ALTER ANY VIEW system privilege
- ALTER DATABASE system privilege
- ALTER DATATYPE system privilege
- BACKUP DATABASE system privilege
- CHANGE PASSWORD system privilege
- CHECKPOINT system privilege
- COMMENT ANY OBJECT system privilege
- CREATE ANY INDEX system privilege
- CREATE ANY MATERIALIZED VIEW system privilege
- CREATE ANY OBJECT system privilege
- CREATE ANY PROCEDURE system privilege
- CREATE ANY SEQUENCE system privilege
- CREATE ANY TABLE system privilege
- CREATE ANY TEXT CONFIGURATION system privilege
- CREATE ANY TRIGGER system privilege
- CREATE ANY VIEW system privilege
- CREATE DATATYPE system privilege
- CREATE EXTERNAL REFERENCE system privilege
- CREATE MATERIALIZED VIEW system privilege
- CREATE MESSAGE system privilege
- CREATE PROCEDURE system privilege
- CREATE PROXY TABLE system privilege
- CREATE TABLE system privilege
- CREATE TEXT CONFIGURATION system privilege
- CREATE VIEW system privilege
- DEBUG ANY PROCEDURE system privilege
- DELETE ANY TABLE system privilege
- DROP ANY INDEX system privilege
- DROP ANY MATERIALIZED VIEW system privilege
- DROP ANY OBJECT system privilege
- DROP ANY PROCEDURE system privilege
- DROP ANY SEQUENCE system privilege
- DROP ANY TABLE system privilege

SQL Statements

- DROP ANY TEXT CONFIGURATION system privilege
- DROP ANY VIEW system privilege
- DROP CONNECTION system privilege
- DROP DATATYPE system privilege
- DROP MESSAGE system privilege
- EXECUTE ANY PROCEDURE system privilege
- LOAD ANY TABLE system privilege
- INSERT ANY TABLE system privilege
- MANAGE ANY DBSPACE system privilege
- MANAGE ANY EVENT system privilege
- MANAGE ANY EXTERNAL ENVIRONMENT system privilege
- MANAGE ANY EXTERNAL OBJECT system privilege
- MANAGE ANY LDAP SERVER system privilege
- MANAGE ANY LOGIN POLICY system privilege
- MANAGE ANY MIRROR SERVER system privilege
- MANAGE ANY OBJECT PRIVILEGES system privilege
- MANAGE ANY SPATIAL OBJECT system privilege
- MANAGE ANY STATISTICS system privilege
- MANAGE ANY USER system privilege
- MANAGE ANY WEB SERVICE system privilege
- MANAGE AUDITING system privilege
- MANAGE MULTIPLEX system privilege
- MANAGE PROFILING system privilege
- MANAGE REPLICATION system privilege
- MANAGE ROLES system privilege
- MONITOR system privilege
- READ CLIENT FILE system privilege
- READ FILE system privilege
- REORGANIZE ANY OBJECT system privilege
- SELECT ANY TABLE system privilege
- SERVER OPERATOR system privilege
- SET ANY PUBLIC OPTION system privilege
- SET ANY SECURITY OPTION system privilege
- SET ANY SYSTEM OPTION system privilege
- SET ANY USER DEFINED OPTION system privilege
- SET USER system privilege (granted with ADMIN ONLY clause)
- TRUNCATE ANY TABLE system privilege
- UPDATE ANY TABLE system privilege
- UPGRADE ROLE system privilege

- USE ANY SEQUENCE system privilege
- VALIDATE ANY OBJECT system privilege
- WRITE CLIENT FILE system privilege
- WRITE FILE system privilege

GRANT USAGE ON SEQUENCE Statement

Grants the USAGE system privilege on a specified sequence to a user or role.

Syntax

```
GRANT USAGE ON SEQUENCE sequence-name
  TO userID [,...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL – syntax is a Persistent Stored Module feature.
- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires one of:

- MANAGE ANY OBJECT PRIVILEGE system privilege.
- You own the sequence.

IF Statement

Lets you conditionally execute the first list of SQL statements whose *search-condition* evaluates to TRUE.

Syntax

```
IF search-condition THEN statement-list
... [ ELSEIF search-condition THEN statement-list ]...
... [ ELSE statement-list ]
... END IF
```

Examples

- **Example 1** – this procedure illustrates the use of the **IF** statement:

```
CREATE PROCEDURE TopCustomer (OUT TopCompany CHAR(35), OUT
TopValue INT)
BEGIN
    DECLARE err_notfound EXCEPTION
    FOR SQLSTATE '02000' ;
    DECLARE curThisCust CURSOR FOR
    SELECT CompanyName, CAST(      sum(SalesOrderItems.Quantity *
Products.UnitPrice) AS INTEGER) VALUE
    FROM Customers
    LEFT OUTER JOIN SalesOrders
    LEFT OUTER JOIN SalesOrsderItems
    LEFT OUTER JOIN Product
    GROUP BY CompanyName ;

    DECLARE ThisValue INT ;
    DECLARE ThisCompany CHAR(35) ;
    SET TopValue = 0 ;
    OPEN curThisCust ;
    CustomerLoop:
    LOOP
        FETCH NEXT curThisCust
        INTO ThisCompany, ThisValue ;
        IF SQLSTATE = err_notfound THEN
            LEAVE CustomerLoop ;
        END IF ;
        IF ThisValue > TopValue THEN
            SET TopValue = ThisValue ;
            SET TopCompany = ThisCompany ;
        END IF ;
    END LOOP CustomerLoop ;
    CLOSE curThisCust ;
END
```

- **Example 2** – this procedure illustrates the use of the **ELSEIF** statement:

```
BEGIN
    DECLARE X INT;
    SET X = 1;
    IF X = 1 THEN
        PRINT '1';
    ELSEIF X = 2 THEN
        PRINT '2';
    ELSE
        PRINT 'something else';
    ENDIF
END
```

Usage

If no *search-condition* evaluates to TRUE, and an ELSE clause exists, the *statement-list* in the ELSE clause is executed. If no *search-condition* evaluates to TRUE, and there is no ELSE clause, the expression returns a NULL value.

Execution resumes at the first statement after the END IF.

When comparing variables to the single value returned by a **SELECT** statement inside an **IF** statement, you must first assign the result of the **SELECT** to another variable.

Note: Do not confuse the syntax of the **IF** statement with that of the IF expression. You cannot nest the **IF** statement.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—The Transact-SQL **IF** statement has a slightly different syntax.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84

IF Statement [T-SQL]

Provides conditional execution of a Transact-SQL statement, as an alternative to the SAP Sybase IQ **IF** statement.

Syntax

```
IF expression
... statement
... [ ELSE [ IF expression ] statement ]...
```

Examples

- **Example 1** – use of the Transact-SQL **IF** statement:

```
IF (SELECT max(id) FROM sysobjects) < 100
  RETURN
ELSE
  BEGIN
    PRINT 'These are the user-created objects'
    SELECT name, type, id
    FROM sysobjects
```

SQL Statements

```
WHERE id < 100
END
```

- **Example 2** – use of the Transact-SQL **ELSEIF** statement:

```
BEGIN
  DECLARE @X INT
  SET @X = 1
  IF @X = 1
    PRINT '1'
  ELSEIF @X = 2
    PRINT '2'
  ELSE
    PRINT 'something else'
END
```

Usage

The Transact-SQL **IF** conditional and the **ELSE** conditional each control the performance of only a single SQL statement or compound statement (between the keywords **BEGIN** and **END**).

In contrast to the SAP Sybase IQ **IF** statement, the Transact-SQL **IF** statement has no **THEN**. The Transact-SQL version also has no **ELSEIF** or **END IF** keywords.

When comparing variables to the single value returned by a **SELECT** statement inside an **IF** statement, you must first assign the result of the **SELECT** to another variable.

Note: You cannot nest the **IF** statement.

Standards

- SQL—Transact-SQL extension to ISO/ANSI SQL grammar.
- Sybase—Adaptive Server Enterprise supports the Transact-SQL **IF** statement.

Permissions

None

INCLUDE Statement [ESQL]

Includes a file into a source program to be scanned by the SQL source language preprocessor.

Syntax

```
INCLUDE filename
```


Parameters

- **filename** – identifier

Usage

The **INCLUDE** statement is very much like the C preprocessor **#include** directive.

However, the SQL preprocessor reads the given file, inserting its contents into the output C file. Thus, if an include file contains information that the SQL preprocessor requires, it should be included with the Embedded SQL **INCLUDE** statement.

Two file names are specially recognized: **SQLCA** and **SQLDA**. Any C program using Embedded SQL must contain this statement before any Embedded SQL statements:

```
EXEC SQL INCLUDE SQLCA;
```

This statement must appear at a position in the C program where static variable declarations are allowed. Many Embedded SQL statements require variables (invisible to the programmer) which are declared by the SQL preprocessor at the position of the **SQLCA** include statement. The **SQLDA** file must be included if any **SQLDAs** are used.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

INSERT Statement

Inserts a single row or a selection of rows, from elsewhere in the current database, into the table. This command can also insert a selection of rows from another database into the table.

Syntax

Syntax 1

```
INSERT [ INTO ] [ owner.]table-name [ ( column-name [, ...] ) ]
    ... VALUES ( [ expression | DEFAULT, ... ] )
or
INSERT [ INTO ] [ owner.]table-name DEFAULT VALUES
```

Syntax 2

```
INSERT [ INTO ] [ owner.]table-name [ ( column-name [, ...] ) ]
    ... insert-load-options insert-select-load-options
    ... select-statement
```

```

insert-load-options:
  [ LIMIT number-of-rows ]
  [ NOTIFY number-of-rows ]
  [ SKIP number-of-rows ]

insert-select-load-options:
  [ WORD SKIP number ]
  [ IGNORE CONSTRAINT constraint-type [, ...] ]
  [ MESSAGE LOG 'string' ROW LOG 'string' [ ONLY LOG logwhat
  [, ...] ] ]
  [ LOG DELIMITED BY 'string' ]

constraint-type:
  { CHECK integer
  | UNIQUE integer
  | NULL integer
  | FOREIGN KEY integer
  | DATA VALUE integer
  } ALL integer
  }

logwhat:
  { CHECK
  | ALL
  | NULL
  | UNIQUE
  | DATA VALUE
  | FOREIGN KEY
  | WORD
  }

```

Syntax 3

```

INSERT [ INTO ] [ owner.]table-name[ ( column-name [, ...] ) ]
  ... insert-load-options insert-select-load-options
  LOCATION 'servername.dbname'
  [ location-options ]
  ... { { select-statement } | 'select statement' }

insert-load-options:
  [ LIMIT number-of-rows ]
  [ NOTIFY number-of-rows ]
  [ SKIP number-of-rows ]

insert-select-load-options:      [ WORD SKIP number ]
  [ IGNORE CONSTRAINT constraint-type [, ...] ]
  [ MESSAGE LOG 'string' ROW LOG 'string' [ ONLY LOG logwhat
  [, ...] ] ]
  [ LOG DELIMITED BY 'string' ]

constraint-type:
  { CHECK integer
  | UNIQUE integer
  | NULL integer
  | FOREIGN KEY integer
  | DATA VALUE integer
  }

```

```

    } ALL integer
  }

logwhat:
{ CHECK
| ALL
| NULL
| UNIQUE
| DATA VALUE
| FOREIGN KEY
| WORD
}

location-options:
[ ENCRYPTED PASSWORD ]
[ PACKETSIZE packet-size ]
[ QUOTED_IDENTIFIER { ON | OFF } ]
[ ISOLATION LEVEL { READ UNCOMMITTED | READ
COMMITTED | SERIALIZABLE } ]

```

Parameters

- **insert-load-options** – options that constrain the load:
 - **LIMIT** – specifies the maximum number of rows to insert into the table from a query. The default is 0 for no limit. The maximum is 2GB -1.
 - **NOTIFY** – specifies that you be notified with a message each time the number of rows are successfully inserted into the table. The default is every 100,000 rows.
 - **SKIP** – defines the number of rows to skip at the beginning of the input tables for this insert. The default is 0.
 - **WORD SKIP** – allows the load to continue when it encounters data longer than the limit specified when the word index was created. The *number* parameter specifies the number of times to ignore the error. Setting this option to 0 means there is no limit.

If a row is not loaded because a word exceeds the maximum permitted size, a warning is written to the `.iqmsg` file. WORD size violations can be optionally logged to the MESSAGE LOG file. If the option is not specified, the operation rolls back on the first occurrence of a word that is longer than the specified limit.

- **IGNORE CONSTRAINT** – determines whether the load engine ignores CHECK, UNIQUE, NULL, DATA VALUE, and FOREIGN KEY integrity constraint violations that occur during a load and the maximum number of violations to ignore before initiating a rollback.

If *limit* is zero, the number of CHECK constraint violations to ignore is infinite. If CHECK is not specified, the first occurrence of any CHECK constraint violation causes the load to roll back. If *limit* is nonzero, then the *limit*+1 occurrence of a CHECK constraint violation causes the load to roll back

- **MESSAGE LOG** – specifies the file names where the load engine logs integrity constraint violations. Timestamps indicating the start and completion of the load are logged in both the MESSAGE LOG and the ROW LOG files. Both MESSAGE LOG and ROW LOG must be specified, or no information about integrity violations is logged.

Information is logged on all integrity constraint-type violations specified in the ONLY LOG clause or for all word index-length violations if the keyword WORD is specified. If the ONLY LOG clause is not specified, no information on integrity constraint violations is logged. Only the timestamps indicating the start and completion of the load are logged.

- **LOG DELIMITED BY** – specifies the separator between data values in the ROW LOG file. The default separator is a comma.
- **ENCRYPTED PASSWORD** – specifies the use of Open Client Library default password encryption when connecting to a remote server. If you specify this parameter and the remote server does not support Open Client Library default password encryption, an error is reported indicating that an invalid user ID or password was used.

To enable the SAP Sybase IQ server to accept a jConnect connection with an encrypted password, set the jConnect ENCRYPT_PASSWORD connection property to true.

- **PACKETSIZE** – specifies the TDS packet-size in bytes. The default TDS packet-size on most platforms is 512 bytes. If the packet size is not specified or is specified as zero, then the default packet size value for the platform is used.

The packet-size value must be a multiple of 512, either equal to the default network packet size or between the default network packet size and the maximum network packet size. The maximum network packet size and the default network packet size are multiples of 512 in the range 512 – 524288 bytes. The maximum network packet size is always greater than or equal to the default network packet size.

- **QUOTED_IDENTIFIER** – sets the QUOTED_IDENTIFIER option on the remote server. The default setting is 'OFF.' You set QUOTED_IDENTIFIER to 'ON' only if any of the identifiers in the SELECT statement are enclosed in double quotes, as in this example using 'c1':

```
INSERT INTO foo
LOCATION 'ase.database'
QUOTED_IDENTIFIER ON {select "c1" from xxx};
```

- **ISOLATION LEVEL** – specifies an isolation level for the connection to a remote server:

Isolation Level	Characteristics
READ UNCOMMITTED	<ul style="list-style-type: none"> • Isolation level 0 • Read permitted on row with or without write lock • No read locks are applied • No guarantee that concurrent transaction will not modify row or roll back changes to row

Isolation Level	Characteristics
READ COMMITTED	<ul style="list-style-type: none"> • Isolation level 1 • Read only permitted on row with no write lock • Read lock acquired and held for read on current row only, but released when cursor moves off the row • No guarantee that data will not change during transaction
SERIALIZABLE	<ul style="list-style-type: none"> • Isolation level 3 • Read only permitted on rows in result without write lock • Read locks acquired when cursor is opened and held until transaction ends

Note: For additional information on the insert-select-load-options and location-options as well as the constraint-type and logwhat parameters, see the LOAD TABLE Statement.

Examples

- **Example 1** – add an Eastern Sales department to the database:

```
INSERT INTO Departments
(DepartmentID, DepartmentName, DepartmentHeadID)
VALUES (600, 'Eastern Sales', 501)
```

- **Example 2** – fill the table dept_head with the names of department heads and their departments:

```
INSERT INTO dept_head (name, dept)
  NOTIFY 20
  SELECT Surname || ' ' || GivenName
  AS name,
  dept_name
FROM Employees JOIN Departments
  ON EmployeeID= DepartmentHeadID
```

- **Example 3** – insert data from the l_shipdate and l_orderkey columns of the lineitem table from the SAP Sybase IQ database iqdet on the remote server detroit into the corresponding columns of the lineitem table in the current database:

```
INSERT INTO lineitem
(l_shipdate, l_orderkey)
  LOCATION 'detroit.iqdet'
  PACKETSIZE 512
  ' SELECT l_shipdate, l_orderkey
FROM lineitem '
```

- **Example 4** – the INSERT statement permits a list of values allowing several rows to be inserted at once.

```
INSERT into t1 values( 10, 20, 30 ), ( 11, 21, 31 ), ( 12, 22,
32 )
```

Usage

Syntax 1 allows the insertion of a single row with the specified expression values. If the list of column names is not specified, the values are inserted into the table columns in the order they were created (the same order as retrieved with **SELECT ***). The row is inserted into the table at an arbitrary position. (In relational databases, tables are not ordered.)

Syntax 2 allows the user to perform a mass insertion into a table using the results of a fully general **SELECT** statement. Insertions are done in an arbitrary order unless the **SELECT** statement contains an **ORDER BY** clause. The columns from the select list are matched ordinally with the columns specified in the column list, or sequentially in the order in which the columns were created.

Note: The **NUMBER(*)** function is useful for generating primary keys with Syntax 2 of the **INSERT** statement.

Syntax 3 **INSERT...LOCATION** is a variation of Syntax 2 that allows you to insert data from an Adaptive Server Enterprise or SAP Sybase IQ database. The *servername.dbname* specified in the **LOCATION** clause identifies the remote server and database for the table in the **FROM** clause. To use Syntax 3, the Adaptive Server Enterprise or SAP Sybase IQ remote server to which you are connecting must exist in the SAP Sybase Open Client *interfaces* or *sql.ini* file on the local machine.

In queries using Syntax 3, you can insert a maximum of 2147483647 rows.

The **SELECT** statement can be delimited by either curly braces or straight single quotation marks.

Note: Curly braces represent the start and end of an escape sequence in the ODBC standard, and might generate errors in the context of ODBC or Sybase Control Center. The workaround is to use single quotes to escape the **SELECT** statement.

The local SAP Sybase IQ server connects to the server and database you specify in the **LOCATION** clause. The results from the queries on the remote tables are returned and the local server inserts the results in the current database. If you do not specify a server name in the **LOCATION** clause, SAP Sybase IQ ignores any database name you specify, since the only choice is the current database on the local server.

When SAP Sybase IQ connects to the remote server, **INSERT...LOCATION** uses the remote login for the user ID of the current connection, if a remote login has been created with **CREATE EXTERNLOGIN** and the remote server has been defined with a **CREATE SERVER** statement. If the remote server is not defined, or if a remote login has not been created for the user ID of the current connection, SAP Sybase IQ connects using the user ID and password of the current connection.

Note: If you rely on the user ID and password of the current connection, and a user changes the password, you must stop and restart the server before the new password takes effect on the

remote server. Remote logins created with **CREATE EXTERNLOGIN** are unaffected by changes to the password for the default user ID.

Creating a remote login with the **CREATE EXTERNLOGIN** statement and defining a remote server with a **CREATE SERVER** statement sets up an external login and password for **INSERT...LOCATION** such that any user can use the login and password in any context. This avoids possible errors due to inaccessibility of the login or password, and is the recommended way to connect to a remote server.

For example, user `russid` connects to the SAP Sybase IQ database and executes this statement:

```
INSERT local_SQL_Types LOCATION 'ase1.ase1db'
{SELECT int_col FROM SQL_Types};
```

On server `ase1`, there exists user ID `ase1user` with password `sybase`. The owner of the table `SQL_Types` is `ase1user`. The remote server is defined on the IQ server as:

```
CREATE SERVER ase1 CLASS 'ASEJDBC'
USING 'system1:4100';
```

The external login is defined on the IQ server as:

```
CREATE EXTERNLOGIN russid TO ase1 REMOTE LOGIN ase1user IDENTIFIED BY
sybase;
```

INSERT...LOCATION connects to the remote server `ase1` using the user ID `ase1user` and the password `sybase` for user `russid`.

Use the **ENCRYPTED PASSWORD** parameter to specify the use of Open Client Library default password encryption when connecting to a remote server. If **ENCRYPTED PASSWORD** is specified and the remote server does not support Open Client Library default password encryption, an error is reported indicating that an invalid user ID or password was used.

When used as a remote server, SAP Sybase IQ supports TDS password encryption. The SAP Sybase IQ server accepts a connection with an encrypted password sent by the client. For information on connection properties to set for password encryption, see *Software Developer's Kit 15.5 > Open Client Client-Library/C Reference Manual > Client-Library Topics > Security features > Adaptive Server Enterprise security features > Security handshaking: encrypted password* for Open Server 15.5.

Note: Password encryption requires Open Client 15.0. TDS password encryption requires Open Client 15.0 ESD #7 or later.

When **INSERT...LOCATION** is transferring data between an SAP Sybase IQ server and a remote SAP Sybase IQ or Adaptive Server Enterprise server, the value of the **INSERT...LOCATION TDS PACKETSIZE** parameter is always 512 bytes, even if you specify a different value for **PACKETSIZE**.

Note: If you specify an incorrect packet size (for example 933, which is not a multiple of 512), the connection attempt fails with an Open Client `ct_connect` "Connection failed" error. Any

unsuccessful connection attempt returns a generic “Connection failed” message. The Adaptive Server Enterprise error log might contain more specific information about the cause of the connection failure.

SAP Sybase IQ does not support the Adaptive Server Enterprise data type `TEXT`, but you can execute **INSERT...LOCATION** (Syntax 3) from both an IQ `CHAR` or `VARCHAR` column whose length is greater than 255 bytes, and from an ASE database column of data type `TEXT`. ASE `TEXT` and `IMAGE` columns can be inserted into columns of other SAP Sybase IQ data types, if SAP Sybase IQ supports the internal conversion. By default, if a remote data column contains over 2GB, SAP Sybase IQ silently truncates the column value to 2GB.

Warning! SAP Sybase IQ does not support the Adaptive Server Enterprise data types `UNICHAR`, `UNIVARCHAR`, or `UNITEXT`. An **INSERT...LOCATION** command from `UNICHAR` or `UNITEXT` to `CHAR` or `CLOB` columns in the `ISO_BINENG` collation may execute without error; if this happens, the data in the columns may be inconsistent. An error is reported in this situation, only if the conversion fails.

Users must be specifically licensed to use the large object functionality of the Unstructured Data Analytics Option.

Note: If you use **INSERT...LOCATION** to insert data selected from a **VARBINARY** column, set **ASE_BINARY_DISPLAY** to `OFF` on the remote database.

INSERT...LOCATION (Syntax 3) does not support the use of variables in the **SELECT** statement.

Inserts can be done into views, provided the **SELECT** statement defining the view has only one table in the **FROM** clause and does not contain a **GROUP BY** clause, an aggregate function, or involve a **UNION** operation.

Character strings inserted into tables are always stored in the case they are entered, regardless of whether the database is case-sensitive or not. Thus, a string “Value” inserted into a table is always held in the database with an uppercase V and the remainder of the letters lowercase. **SELECT** statements return the string as 'Value.' If the database is not case-sensitive, however, all comparisons make 'Value' the same as 'value,' 'VALUE,' and so on. Further, if a single-column primary key already contains an entry Value, an **INSERT** of value is rejected, as it would make the primary key not unique.

Whenever you execute an **INSERT...LOCATION** statement, SAP Sybase IQ loads the localization information needed to determine language, collation sequence, character set, and date/time format. If your database uses a nondefault locale for your platform, you must set an environment variable on your local client to ensure that SAP Sybase IQ loads the correct information.

If you set the `LC_ALL` environment variable, SAP Sybase IQ uses its value as the locale name. If `LC_ALL` is not set, SAP Sybase IQ uses the value of the `LANG` environment variable. If neither variable is set, SAP Sybase IQ uses the default entry in the locales file.

Use the (DEFAULT), DEFAULT VALUES or VALUES () clauses to insert rows with all default values. Assuming that there are 3 columns in table t2, these examples are semantically equivalent:

```
INSERT INTO t2 values (DEFAULT, DEFAULT, DEFAULT);
```

```
INSERT INTO t2 DEFAULT VALUES;
```

```
INSERT INTO t2() VALUES();
```

INSERT...VALUES also supports multiple rows. The following example inserts 3 rows into table t1:

```
CREATE TABLE t1(c1 varchar(30));
INSERT INTO t1 VALUES ('morning'),('afternoon'),
('evening');
```

SAP Sybase IQ treats all load/inserts as full-width inserts. Columns not explicitly specified on the load/insert statement, the value loaded will either be the column's DEFAULT value (if one is defined) or NULL (if no DEFAULT value is defined for the column).

SAP Sybase IQ supports column DEFAULT values for **INSERT...VALUES**, **INSERT...SELECT**, and **INSERT...LOCATION**. If a DEFAULT value is specified for a column, this DEFAULT value is used as the value of the column in any **INSERT** (or **LOAD**) statement that does not specify a value for the column.

An **INSERT** from a stored procedure or function is not permitted, if the procedure or function uses **COMMIT**, **ROLLBACK**, or some **ROLLBACK TO SAVEPOINT** statements.

The result of a **SELECT...FROM** may be slightly different from the result of an **INSERT...SELECT...FROM** due to an internal data conversion of an imprecise data type, such as DOUBLE or NUMERIC, for optimization during the insert. If a more precise result is required, a possible workaround is to declare the column as a DOUBLE or NUMERIC data type with a higher precision.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Adaptive Server Enterprise (excluding the *insert-load-options*).

Permissions

Requires INSERT privilege on the table.

See also

- *CREATE EXTERNLOGIN Statement* on page 131
- *DELETE Statement* on page 248
- *LOAD TABLE Statement* on page 343

INSTALL JAVA Statement

Makes Java classes available for use within a database.

Syntax

```
INSTALL JAVA [ install-mode ] [ JAR jar-name ]
FROM source
```

```
install-mode:
{ NEW | UPDATE }
```

```
source:
{ FILE file-name | URL url-value }
```

Parameters

- **NEW** – (default) requires that the referenced Java classes be new classes, rather than updates of currently installed classes. An error occurs if a class with the same name exists in the database and the **NEW** install mode clause is used
- **UPDATE** – An install mode of specifies that the referenced Java classes may include replacements for Java classes already installed in the given database.
- **JAR** – a character string value of up to 255 bytes that is used to identify the retained JAR in subsequent **INSTALL**, **UPDATE**, and **REMOVE** statements. *jar-name* or text-pointer must designate a JAR file or a column containing a JAR. JAR files typically have extensions of *.jar* or *.zip*.

Installed JAR and zip files can be compressed or uncompressed. However, JAR files produced by the Sun JDK **jar** utility are not supported. Files produced by other zip utilities are supported.

If the **JAR** option is specified, then the JAR is retained as a JAR after the classes that it contains have been installed. That JAR is the associated JAR of each of those classes. The set of JARs installed in a database with the **JAR** clause are called the retained JARs of the database.

Retained JARs are referenced in **INSTALL** and **REMOVE** statements. Retained JARs have no effect on other uses of Java-SQL classes. Retained JARs are used by the SQL system for requests by other systems for the class associated with given data. If a requested class has an associated JAR, the SQL system can supply that JAR, rather than the individual class.

- **source** – specifies the location of the Java classes to be installed and must identify either a class file or a JAR file.

The formats supported for *file-name* include fully qualified file names, such as 'c:\libs\jарname.jar' and '/usr/u/libs/jарname.jar', and relative file names, which are relative to the current working directory of the database server.

The class definition for each class is loaded by the VM of each connection the first time that class is used. When you **INSTALL** a class, the VM on your connection is implicitly restarted. Therefore, you have immediate access to the new class, whether the **INSTALL** uses an install-mode clause of **NEW** or **UPDATE**.

For other connections, the new class is loaded the next time a VM accesses the class for the first time. If the class is already loaded by a VM, that connection does not see the new class until the VM is restarted for that connection (for example, with a **STOP JAVA** and **START JAVA**).

Examples

- **Example 1** – install the user-created Java class named “Demo” by providing the file name and location of the class:

```
INSTALL JAVA NEW
FROM FILE 'D:\JavaClass\Demo.class'
```

After installation, the class is referenced using its name. Its original file path location is no longer used. For example, this statement uses the class installed in the previous statement:

```
CREATE VARIABLE d Demo
```

If the Demo class was a member of the package sybase.work, the fully qualified name of the class must be used:

```
CREATE VARIABLE d sybase.work.Demo
```

- **Example 2** – install all the classes contained in a zip file and associate them within the database with a JAR file name:

```
INSTALL JAVA
JAR 'Widgets'
FROM FILE 'C:\Jars\Widget.zip'
```

The location of the zip file is not retained and classes must be referenced using the fully qualified class name (package name and class name).

Usage

.Only new connections established after installing the class, or that use the class for the first time after installing the class, use the new definition. Once the Java VM loads a class definition, it stays in memory until the connection closes.

If you have been using a Java class or objects based on a class in the current connection, you need to disconnect and reconnect to use the new class definition.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

- Requires the `MANAGE ANY EXTERNAL OBJECT` system privilege and a newer version of the compiled class file or JAR file available in a file on disk.
- All installed classes can be referenced in any way by any user.

See also

- *REMOVE Statement* on page 387

IQ UTILITIES Statement

Starts a cache monitor that collects buffer cache statistics.

Syntax

```
IQ UTILITIES { MAIN | PRIVATE }  
  [ INTO ] table-name  
  { START MONITOR ['monitor-options']  
  | STOP MONITOR }
```

monitor-options:

```
{ -summary  
  | {-append | -truncate } -bufalloc  
  | -cache  
  | -cache_by_type  
  | -contention  
  | -debug  
  | -file_suffix suffix  
  | -io  
  | -interval seconds  
  | -threads }...
```

Parameters

- **START MONITOR** – starts the IQ buffer cache monitor.
- **MAIN** – monitors all tables in the main buffer cache of the IQ Store.
- **PRIVATE** – monitors all tables in the temp buffer cache of the temporary Store.
- **dummy_table_name** – can be any SAP Sybase IQ base or temporary table. The table name is required for syntactic compatibility with other **IQ UTILITIES** commands. It is best to have a table that you use only for monitoring.

- **monitor_options** – controls buffer cache monitor output. You can specify more than one, and they must be enclosed with quotation marks.

Option	Description
-summary	Displays summary information for both the main and temp buffer caches. If you do not specify any monitor options, you receive a summary report. Usage: <code>monitor_options -summary</code>
-cache	Displays main or temp buffer cache activity in detail. Critical fields are Finds, HR%, and Bwaits. Usage: <code>monitor_options -cache</code>
-cache_by_type	Breaks -cache results down by IQ page type. (An exception is the Bwaits column, which shows a total only.) This format is most useful when you need to supply information to Technical Support. Usage: <code>monitor_options -cache_by_type</code>
-file_suffix	Creates a monitor output file named <dbname>.<connid>-<main_or_temp>-<suffix>. If you do not specify an optional file extension, the file extension defaults to .iqmon. Usage: <code>monitor_options -file_suffix {extension}</code>
-io	Displays main or temp (private) buffer cache I/O rates and compression ratios during the specified interval. These counters represent all activity for the server; the information is not broken out by device. Usage: <code>monitor_options -io</code>
-bufalloc	Displays information on the main or temp buffer allocator, which reserves space in the buffer cache for objects like sorts, hashes, and bitmaps. Usage: <code>monitor_options -bufalloc</code>
-contention	Displays many key buffer cache and memory manager locks. These lock and mutex counters show the activity within the buffer cache and heap memory and how quickly these locks were resolved. Timeout numbers that exceed 20% indicate a problem. Usage: <code>monitor_options -contention</code>
-threads	Displays the processing thread manager counts. Values are server-wide (i.e., it does not matter whether you select this option for main or private). Usage: <code>monitor_options -threads</code>

Option	Description
-interval	<p>Specifies the reporting interval in seconds. The default is every 60 seconds. The minimum is every 2 seconds.</p> <p>You can usually get useful results by running the monitor at the default interval during a query or time of day with performance problems. Short intervals may not give meaningful results. Intervals should be proportional to the job time; one minute is generally more than enough. Usage:</p> <pre>monitor_options -interval</pre>
-append -truncate	<p>Append or truncate output to existing output file. Truncate is the default. Usage:</p> <pre>monitor_options -append -truncate</pre>
-debug	<p>Displays all information available to the performance monitor, whether or not there is a standard display mode that covers the same information. -debug is used mainly to supply information to Technical Support. Usage:</p> <pre>monitor_options -debug</pre>

- **STOP MONITOR** – similar to **START MONITOR** except that you do not need to specify any options:

Note:

- To simplify monitor use, create a stored procedure to declare the dummy table, specify its output location, and start the monitor.
 - The interval, with two exceptions, applies to each line of output, not to each page. The exceptions are the -cache_by_type and -debug clauses, where a new page begins for each display.
-

Examples

- **Example 1** – start the buffer cache monitor and record activity for the IQ temp buffer cache:

```
IQ UTILITIES PRIVATE INTO monitor START MONITOR '-cache -interval 20'
```

Usage

Issue separate commands to monitor each buffer cache. Keep each sessions open while the monitor collects results; a monitor run stops when you close its connection. A connection can run up to a maximum of two monitor runs, one for the main and one for the temp buffer cache.

To control the directory placement of monitor output files, set the **MONITOR_OUTPUT_DIRECTORY** option. If this option is not set, the monitor sends

output to the same directory as the database. All monitor output files are used for the duration of the monitor runs. They remain after a monitor run has stopped.

Either declare a temporary table for use in monitoring, or create a permanent dummy table when you create a new database, before creating any multiplex query servers. These solutions avoid DDL changes, so that data stays up on query servers during production runs.

On UNIX-like operating systems, you can watch monitor output as queries are running. For example:

Starting the monitor with this command:

```
iq utilities main into monitor_tab
start monitor "-cache -interval 2 -file_suffix iqmon"
```

sends the output to an ASCII file with the name `dbname.conn#[main|temp]-iqmon`. So, for the `iqdemo` database, the buffer monitor would send the results to `iqdemo.2-main-iqmon`

The buffer cache monitor writes the results of each run to these logs:

- `dbname.connection#-main-iqmon` //for main buffer cache results
- `dbname.connection#-temp-iqmon` //for temp buffer cache results

The prefix `dbname.connection#` represents your database name and connection number. If you see more than one connection number and are uncertain which is yours, you can run the Catalog stored procedure `sa_conn_info`. This procedure displays the connection number, user ID, and other information for each active connection to the database. The `-file_suffic` clause to change the suffix `iqmon` to a suffix of your choice. Use a text editor to display or print a file. Running the monitor again from the same database and connection number, overwrites the previous results. To save the results of a monitor run, copy the file to another location or use the `-append` option.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported in Adaptive Server Enterprise.

Permissions

None

See also

- *MONITOR_OUTPUT_DIRECTORY* Option on page 587

LEAVE Statement

Continues execution by leaving a compound statement or **LOOP**.

Syntax

```
LEAVE statement-label
```

Examples

- **Example 1** – this code fragment shows how to use the **LEAVE** statement to leave a loop:

```
SET i = 1;
lbl:
LOOP
  INSERT
  INTO Counters ( number )
  VALUES ( i ) ;
  IF i >= 10 THEN
    LEAVE lbl ;
  END IF ;
  SET i = i + 1
END LOOP lbl
```

- **Example 2** – this code fragment uses **LEAVE** in a nested loop:

```
outer_loop:
LOOP
  SET i = 1;
  inner_loop:
  LOOP
    ...
    SET i = i + 1;
    IF i >= 10 THEN
      LEAVE outer_loop
    END IF
  END LOOP inner_loop
END LOOP outer_loop
```

Usage

LEAVE is a control statement that lets you leave a labeled compound statement or a labeled loop. Execution resumes at the first statement after the compound statement or loop.

The compound statement that is the body of a procedure has an implicit label that is the same as the name of the procedure.

Standards

- SQL—ISO/ANSI SQL compliant.

- Sybase—Not supported in Adaptive Server Enterprise. The break statement provides a similar feature for Transact-SQL compatible procedures.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84
- *FOR Statement* on page 286
- *LOOP Statement* on page 364

LOAD TABLE Statement

Imports data into a database table from an external file.

Syntax

```
LOAD [ INTO ] TABLE [ owner.]table-name
... ( load-specification [, ...] )
... { FROM | USING [ CLIENT ] FILE }
{ 'filename-string' | filename-variable } [, ...]
... [ CHECK CONSTRAINTS { ON | OFF } ]
... [ DEFAULTS { ON | OFF } ]
... [ QUOTES OFF ]
... ESCAPES OFF
... [ FORMAT { ascii | binary | bcp } ]
... [ DELIMITED BY 'string' ]
... [ STRIP { OFF | RTRIM } ]
... [ WITH CHECKPOINT { ON | OFF } ]
... [ BYTE ORDER { NATIVE | HIGH | LOW } ]
... [ LIMIT number-of-rows ]
... [ NOTIFY number-of-rows ]
... [ ON FILE ERROR { ROLLBACK | FINISH | CONTINUE } ]
... [ PREVIEW { ON | OFF } ]
... [ ROW DELIMITED BY 'delimiter-string' ]
... [ SKIP number-of-rows ]
... [ HEADER SKIP number [ HEADER DELIMITED BY 'string' ] ]
... [ WORD SKIP number ]
... [ ON PARTIAL INPUT ROW { ROLLBACK | CONTINUE } ]
... [ IGNORE CONSTRAINT constraint-type [, ...] ]
... [ MESSAGE LOG 'string' ROW LOG 'string' [ ONLY LOG log-what
[, ...] ]
... [ LOG DELIMITED BY 'string' ]

load-specification:
{ column-name [ column-spec ]
  | FILLER ( filler-type ) }

column-spec:
{ ASCII ( input-width )
```

```

| BINARY [ WITH NULL BYTE ]
| PREFIX { 1 | 2 | 4 }
| 'delimiter-string'
| DATE [ FORMAT ] ( input-date-format ) [, input-date-format, ...]
| DATETIME [ FORMAT ] ( input-datetime-format [, input-datetime-
format, ...] )
| ENCRYPTED ( data-type 'key-string' [, 'algorithm-string' ] )
| DEFAULT default-value }
[ NULL ( { BLANKS | ZEROS | 'literal', ...} )

filler-type:
{ input-width
| PREFIX { 1 | 2 | 4 }
| 'delimiter-string'
}

constraint-type:
{ CHECK integer
| UNIQUE integer
| NULL integer
| FOREIGN KEYinteger
| DATA VALUE integer
| ALL integer
}

log-what:
{ CHECK
| ALL
| NULL
| UNIQUE
| DATA VALUE
| FOREIGN KEY
| WORD
}

```

Parameters

- **FROM** – identifies one or more files from which to load data. To specify more than one file, use a comma to separate each *filename-string*. The *filename-string* is passed to the server as a string. The string is therefore subject to the same formatting requirements as other SQL strings.

To indicate directory paths on Windows, the backslash character \ must be represented by two backslashes. Therefore, the statement to load data from the file `c:\temp\input.dat` into the `Employees` table is:

```
LOAD TABLE Employees
FROM 'c:\\temp\\input.dat' ...
```

The path name is relative to the database server, not to the client application. If you are running the statement on a database server on some other computer, the directory names refers to directories on the server machine, not on the client machine. When loading a multiplex database, use absolute (fully qualified) paths in all file names. Do not use relative path names.

Because of resource constraints, SAP Sybase IQ does not guarantee that all the data can be loaded. If resource allocation fails, the entire load transaction is rolled back. The files are read one at a time, and processed in the order specified in the FROM clause. Any SKIP or LIMIT value only applies in the beginning of the load, not for each file.

The LOAD TABLE FROM clause is deprecated, but may be used to specify a file that exists on the server. This example loads data from the file a.inp on a client computer.

```
LOAD TABLE t1(c1,c2,filler(30))
USING CLIENT FILE 'c:\\client-data\\a.inp'
QUOTES OFF ESCAPES OFF
IGNORE CONSTRAINT UNIQUE 0, NULL 0
MESSAGE LOG 'c:\\client-data\\m.log'
ROW LOG 'c:\\client-data\\r.log'
ONLY LOG UNIQUE
```

- **USING** – USING FILE loads one or more files from the server. This clause is synonymous with specifying the FROM *filename* clause. USING CLIENT FILE bulk loads one or more files from a client. The character set of the file on the client side must be the same as the server collation. SAP Sybase IQ serially processes files in the file list. Each file is locked in read mode as it is processed, then unlocked. Client-side bulk loading incurs no administrative overhead, such as extra disk space, memory or network-monitoring daemon requirements, but does force single threaded processing for each file.

When bulk loading large objects, the USING CLIENT FILE clause applies to both primary and secondary files.

During client-side loads, the IGNORE CONSTRAINT log files are created on the client host and any error while creating the log files causes the operation to roll back.

Client-side bulk loading is supported by Interactive SQL and ODBC/JDBC clients using the Command Sequence protocol. It is not supported by clients using the TDS protocol. For data security over a network, use Transport Layer Security. To control who can use client-side bulk loads, use the secure feature (-sf) server startup switch, the **ALLOW_READ_CLIENT_FILE** database option, and/or the READCLIENTFILE access control.

- **CHECK CONSTRAINTS** – evaluates check constraints, which you can ignore or log. CHECK CONSTRAINTS defaults to ON.

Setting CHECK CONSTRAINTS OFF causes SAP Sybase IQ to ignore all check constraint violations. This can be useful, for example, during database rebuilding. If a table has check constraints that call user-defined functions that are not yet created, the rebuild fails unless this option is set to OFF.

This option is mutually exclusive to the following options. If any of these options are specified in the same load, an error results:

- IGNORE CONSTRAINT ALL
- IGNORE CONSTRAINT CHECK

- LOG ALL
- LOG CHECK
- **DEFAULTS** – uses a column's default value. This option is ON by default. If the DEFAULTS option is OFF, any column not present in the column list is assigned NULL. The setting for the DEFAULTS option applies to all column DEFAULT values, including AUTOINCREMENT.
- **QUOTES** – indicates that input strings are enclosed in quote characters. QUOTES is an optional parameter and is ON by default. The quote character is either an apostrophe (single quote) or a quotation mark (double quote). The first such character encountered in a string is treated as the quote character for the string. String data must be terminated with a matching quote.

With QUOTES ON, column or row delimiter characters can be included in the column value. Leading and ending quote characters are assumed not to be part of the value and are excluded from the loaded data value.

To include a quote character in a value with QUOTES ON, use two quotes. For example, this line includes a value in the third column that is a single quote character:

```
'123 High Street, Anytown', '(715) 398-2354', ''''
```

With STRIP turned on (the default), trailing blanks are stripped from values before they are inserted. Trailing blanks are stripped only for non-quoted strings. Quoted strings retain their trailing blanks. Leading blank or TAB characters are trimmed only when the setting is ON.

The data extraction facility provides options for handling quotes (**TEMP_EXTRACT_QUOTES**, **TEMP_EXTRACT_QUOTES_ALL**, and **TEMP_EXTRACT_QUOTE**). If you plan to extract data to be loaded into an IQ main store table and the string fields contain column or row delimiter under default ASCII extraction, use the **TEMP_EXTRACT_BINARY** option for the extract and the **FORMAT** binary and **QUOTES OFF** options for **LOAD TABLE**.

Limits:

- QUOTES ON applies only to column-delimited ASCII fields.
- With QUOTES ON, the first character of a column delimiter or row terminator cannot be a single or double quote mark.
- QUOTES ON forces single threaded processing for a given file.
- The QUOTES option does not apply to loading binary large object (BLOB) or character large object (CLOB) data from the secondary file, regardless of its setting. A leading or trailing quote is loaded as part of CLOB data. Two consecutive quotes between enclosing quotes are loaded as two consecutive quotes with the QUOTES ON option.
- Adaptive Server Enterprise BCP does not support the QUOTES option. All field data is copied in or out equivalent to the QUOTES OFF setting. As QUOTES ON is the default

setting for the SAP Sybase IQ **LOAD TABLE** statement, you must specify **QUOTES OFF** when importing ASE data from BCP output to an SAP Sybase IQ table.

Exceptions:

- If **LOAD TABLE** encounters any nonwhite characters after the ending quote character for an enclosed field, this error is reported and the load operation is rolled back:

```
Non-SPACE text found after ending quote character for
an enclosed field.
SQLSTATE: QTA14      SQLCODE: -1005014L
```

- With **QUOTES ON**, if a single or double quote is specified as the first character of the column delimiter, an error is reported and the load operation fails:

```
Single or double quote mark cannot be the 1st character
of column delimiter or row terminator with QUOTES option
ON.
SQLSTATE: QCA90      SQLCODE: -1013090L
```

- **ESCAPES** – if you omit a *column-spec* definition for an input field and **ESCAPES** is **ON** (the default), characters following the backslash character are recognized and interpreted as special characters by the database server. You can include newline characters as the combination `\n`, and other characters as hexadecimal ASCII codes, such as `\x09` for the tab character. A sequence of two backslash characters (`\\`) is interpreted as a single backslash. For SAP Sybase IQ, you must set **ESCAPES OFF**.
- **FORMAT** – SAP Sybase IQ supports ASCII and binary input fields. The format is usually defined by the *column-spec* described above. If you omit that definition for a column, by default SAP Sybase IQ uses the format defined by this option. Input lines are assumed to have **ascii** (the default) or **binary** fields, one row per line, with values separated by the column delimiter character.

SAP Sybase IQ also accepts data from BCP character files as input to the **LOAD TABLE** command.

- The BCP data file loaded into SAP Sybase IQ tables using the **LOAD TABLE FORMAT BCP** statement must be exported (BCP OUT) in cross-platform file format using the `-c` option.
- For **FORMAT BCP**, the default column delimiter for the **LOAD TABLE** statement is `<tab>` and the default row terminator is `<newline>`.
- For **FORMAT BCP**, the last column in a row must be terminated by the row terminator, not by the column delimiter. If the column delimiter is present before the row terminator, then the column delimiter is treated as a part of the data.
- Data for columns that are not the last column in the load specification must be delimited by the column delimiter only. If a row terminator is encountered before a column delimiter for a column that is not the last column, then the row terminator is treated as a part of the column data.

- Column delimiter can be specified via the **DELIMITED BY** clause. For **FORMAT BCP**, the delimiter must be less than or equal to 10 characters in length. An error is returned, if the delimiter length is more than 10.
- For **FORMAT BCP**, the load specification may contain only column names, **NULL**, and **ENCRYPTED**. An error is returned, if any other option is specified in the load specification.

For example, these **LOAD TABLE** load specifications are valid:

```
LOAD TABLE x( c1, c2 null(blanks), c3 )
FROM 'bcp_file.bcp'
FORMAT BCP
...
```

```
LOAD TABLE x( c1 encrypted(bigint,'KEY-ONE','aes'), c2, c3 )
FROM 'bcp_file.bcp'
FORMAT BCP
...
```

- **DELIMITED BY** – if you omit a column delimiter in the *column-spec* definition, the default column delimiter character is a comma. You can specify an alternative column delimiter by providing a single ASCII character or the hexadecimal character representation. The **DELIMITED BY** clause is:

```
... DELIMITED BY '\x09' ...
```

To use the newline character as a delimiter, you can specify either the special combination `\n` or its ASCII value `\x0a`. Although you can specify up to four characters in the *column-spec delimiter-string*, you can specify only a single character in the **DELIMITED BY** clause.

- **STRIP** – determines whether unquoted values should have trailing blanks stripped off before they are inserted. The **LOAD TABLE** command accepts these **STRIP** keywords:
 - **STRIP OFF** – do not strip off trailing blanks.
 - **STRIP RTRIM** – strip trailing blanks.
 - **STRIP ON** – deprecated. Use **STRIP RTRIM**.

With **STRIP** turned on (the default), SAP Sybase IQ strips trailing blanks from values before inserting them. This is effective only for **VARCHAR** data. **STRIP OFF** preserves trailing blanks.

Trailing blanks are stripped only for unquoted strings. Quoted strings retain their trailing blanks. If you do not require blank sensitivity, you can use the **FILLER** option as an alternative to be more specific in the number of bytes to strip, instead of all the trailing spaces. **STRIP OFF** is more efficient for SAP Sybase IQ, and it adheres to the ANSI standard when dealing with trailing blanks. (**CHAR** data is always padded, so the **STRIP** option only affects **VARCHAR** data.)

The **STRIP** option applies only to variable-length non-binary data and does not apply to ASCII fixed-width inserts. For example, assume this schema:

```

CREATE TABLE t( c1 VARCHAR(3) );
LOAD TABLE t( c1 ',' ) ..... STRIP RTRIM // trailing blanks
trimmed

LOAD TABLE t( c1 ',' ) ..... STRIP OFF // trailing blanks
not trimmed

LOAD TABLE t( c1 ASCII(3) ) ... STRIP RTRIM // trailing blanks
not trimmed
LOAD TABLE t( c1 ASCII(3) ) ... STRIP OFF // trailing blanks
trimmed

LOAD TABLE t( c1 BINARY ) ..... STRIP RTRIM // trailing blanks
trimmed
LOAD TABLE t( c1 BINARY ) ..... STRIP OFF // trailing blanks
trimmed

```

Trailing blanks are always trimmed from binary data.

- **WITH CHECKPOINT** – determines whether SAP Sybase IQ performs a checkpoint. This option is useful only when loading SQL Anywhere tables in an SAP Sybase IQ database.

The default setting is OFF. If this clause is set to ON, a checkpoint is issued after successfully completing and logging the statement. If the server fails after a connection commits and before the next checkpoint, the data file used to load the table must be present for the recovery to complete successfully. However, if WITH CHECKPOINT ON is specified, and recovery is subsequently required, the data file need not be present at the time of recovery.

The data files are required, regardless of what is specified for this clause, if the database becomes corrupt and you need to use a backup and apply the current log file.

Warning! If you set the database option `CONVERSION_ERROR` to OFF, you may load bad data into your table without any error being reported. If you do not specify `WITH CHECKPOINT ON`, and the database needs to be recovered, the recovery may fail as `CONVERSION_ERROR` is ON (the default value) during recovery. It is recommended that you do not load tables when `CONVERSION_ERROR` is set to OFF and `WITH CHECKPOINT ON` is not specified.

See also *CONVERSION_ERROR Option [TSQL]*.

- **BYTE ORDER** – specifies the byte order during reads. This option applies to all binary input fields. If none are defined, this option is ignored. SAP Sybase IQ always reads binary data in the format native to the machine it is running on (default is NATIVE). You can also specify:
 - HIGH when multibyte quantities have the high order byte first (for big endian platforms like Sun, IBM AIX, and HP).

- **LOW** when multibyte quantities have the low order byte first (for little endian platforms like Windows).
- **LIMIT** – specifies the maximum number of rows to insert into the table. The default is 0 for no limit. The maximum is $2^{31} - 1$ (2147483647) rows.
- **NOTIFY** – specifies that you be notified with a message each time the specified number of rows is successfully inserted into the table. The default is 0, meaning no notifications are printed. The value of this option overrides the value of the NOTIFY_MODULUS database option.
- **ON FILE ERROR** – specifies the action SAP Sybase IQ takes when an input file cannot be opened because it does not exist or you have incorrect permissions to read the file. You can specify one of the following:
 - **ROLLBACK** – aborts the entire transaction (the default).
 - **FINISH** – finishes the insertions already completed and ends the load operation.
 - **CONTINUE** – returns an error but only skips the file to continue the load operation.Only one ON FILE ERROR clause is permitted.
- **PREVIEW** – displays the layout of input into the destination table including starting position, name, and data type of each column. SAP Sybase IQ displays this information at the start of the load process. If you are writing to a log file, this information is also included in the log
- **ROW DELIMITED BY delimiter-string** – specifies a string up to 4 bytes in length that indicates the end of an input record. You can use this option only if all fields within the row are any of the following:
 - Delimited with column terminators
 - Data defined by the DATE or DATETIME *column-spec* options
 - ASCII fixed length fields

Always include ROW DELIMITED BY to insure parallel loads. Omitting this clause from the LOAD specification may cause SAP Sybase IQ to load serially rather than in parallel.

You cannot use this option if any input fields contain binary data. With this option, a row terminator causes any missing fields to be set to NULL. All rows must have the same row delimiters, and it must be distinct from all column delimiters. The row and field delimiter strings cannot be an initial subset of each other. For example, you cannot specify “*” as a field delimiter and “*#” as the row delimiter, but you could specify “#” as the field delimiter with that row delimiter.

If a row is missing its delimiters, SAP Sybase IQ returns an error and rolls back the entire load transaction. The only exception is the final record of a file where it rolls back that row and returns a warning message. On Windows, a row delimiter is usually indicated by the

newline character followed by the carriage return character. You might need to specify this as the *delimiter-string* (see above for description) for either this option or FILLER.

- **SKIP** – defines the number of rows to skip at the beginning of the input tables for this load. The maximum number of rows to skip is $2^{31} - 1$ (2147483647). The default is 0. SKIP runs in single-threaded mode as it reads the rows to skip.
- **HEADER SKIP...HEADER DELIMITED BY** – specifies a number of lines at the beginning of the data file, including header rows, for **LOAD TABLE** to skip. All **LOAD TABLE** column specifications and other load options are ignored, until the specified number of rows is skipped.
 - The number of lines to skip is greater than or equal to zero.
 - Lines are determined by a 1 to 4 character delimiter string specified in the **HEADER DELIMITED BY** clause. The default **HEADER DELIMITED BY** string is the ‘\n’ character.
 - The **HEADER DELIMITED BY** string has a maximum length of four characters. An error is returned, if the string length is greater than four or less than one.
 - When a non-zero **HEADER SKIP** value is specified, all data inclusive of the **HEADER DELIMITED BY** delimiter is ignored, until the delimiter is encountered the number of times specified in the **HEADER SKIP** clause.
 - All **LOAD TABLE** column specifications and other load options are ignored, until the specified number of rows has been skipped. After the specified number of rows has been skipped, the **LOAD TABLE** column specifications and other load options are applied to the remaining data.
 - The "header" bytes are ignored only at the beginning of the data. When multiple files are specified in the **USING** clause, **HEADER SKIP** only ignores data starting from the first row of the first file, until it skips the specified number of header rows, even if those rows exist in subsequent files. **LOAD TABLE** does not look for headers once it starts parsing actual data.
 - No error is reported, if **LOAD TABLE** processes all input data before skipping the number of rows specified by **HEADER SKIP**.
- **WORD SKIP** – allows the load to continue when it encounters data longer than the limit specified when the word index was created.

If a row is not loaded because a word exceeds the maximum permitted size, a warning is written to the `.iqmsg` file. **WORD** size violations can be optionally logged to the **MESSAGE LOG** file and rejected rows logged to the **ROW LOG** file specified in the **LOAD TABLE** statement.

- If the option is not specified, **LOAD TABLE** reports an error and rolls back on the first occurrence of a word that is longer than the specified limit.
- *number* specifies the number of times the “Words exceeding the maximum permitted word length not supported” error is ignored.

- 0 (zero) means there is no limit.
- **ON PARTIAL INPUT ROW** – specifies the action to take when a partial input row is encountered during a load. You can specify one of the following:
 - CONTINUE issues a warning and continues the load operation. This is the default.
 - ROLLBACK aborts the entire load operation and reports the error.

```
Partial input record skipped at EOF.
SQLSTATE: QDC32      SQLSTATE: -1000232L
```

- **IGNORE CONSTRAINT** – specifies whether to ignore CHECK, UNIQUE, NULL, DATA VALUE, and FOREIGN KEY integrity constraint violations that occur during a load and the maximum number of violations to ignore before initiating a rollback. Specifying each *constrainttype* has the following result:
 - **CHECK limit** – if *limit* specifies zero, the number of CHECK constraint violations to ignore is infinite. If CHECK is not specified, the first occurrence of any CHECK constraint violation causes the **LOAD** statement to roll back. If *limit* is nonzero, then the *limit*+1 occurrence of a CHECK constraint violation causes the load to roll back.
 - **UNIQUE limit** – if *limit* specifies zero, then the number of UNIQUE constraint violations to ignore is infinite. If *limit* is nonzero, then the *limit*+1 occurrence of a UNIQUE constraint violation causes the load to roll back.
 - **NULL limit** – if *limit* specifies zero, then the number of NULL constraint violations to ignore is infinite. If *limit* is nonzero, then the *limit*+1 occurrence of a NULL constraint violation causes the load to roll back.
 - **FOREIGN KEY limit** – if *limit* specifies zero, the number of FOREIGN KEY constraint violations to ignore is infinite. If *limit* is nonzero, then the *limit*+1 occurrence of a FOREIGN KEY constraint violation causes the load to roll back.
 - **DATA VALUE limit** – if the database option `CONVERSION_ERROR = ON`, an error is reported and the statement rolls back. If *limit* specifies zero, then the number of DATA VALUE constraint violations (data type conversion errors) to ignore is infinite. If *limit* is nonzero, then the *limit*+1 occurrence of a DATA VALUE constraint violation causes the load to roll back.
 - **ALL limit** – if the database option `CONVERSION_ERROR = ON`, an error is reported and the statement rolls back. If *limit* specifies zero, then the cumulative total of all integrity constraint violations to ignore is infinite. If *limit* is nonzero, then load rolls back when the cumulative total of all ignored UNIQUE, NULL, DATA VALUE, and FOREIGN KEY integrity constraint violations exceeds the value of *limit*. For example, you specify this IGNORE CONSTRAINT option:

```
IGNORE CONSTRAINT NULL 50, UNIQUE 100, ALL 200
```

The total number of integrity constraint violations cannot exceed 200, whereas the total number of NULL and UNIQUE constraint violations cannot exceed 50 and 100, respectively. Whenever any of these limits is exceeded, the LOAD TABLE statement rolls back.

Note: A single row can have more than one integrity constraint violation. Every occurrence of an integrity constraint violation counts towards the limit of that type of violation.

Set the `IGNORE CONSTRAINT` option limit to a nonzero value if you are logging the ignored integrity constraint violations. Logging an excessive number of violations affects the performance of the load

If `CHECK`, `UNIQUE`, `NULL`, or `FOREIGN KEY` is not specified in the `IGNORE CONSTRAINT` clause, then the load rolls back on the first occurrence of each of these types of integrity constraint violation.

If `DATA VALUE` is not specified in the `IGNORE CONSTRAINT` clause, then the load rolls back on the first occurrence of this type of integrity constraint violation, unless the database option `CONVERSION_ERROR = OFF`. If `CONVERSION_ERROR = OFF`, a warning is reported for any `DATA VALUE` constraint violation and the load continues.

When the load completes, an informational message regarding integrity constraint violations is logged in the `.iqmsg` file. This message contains the number of integrity constraint violations that occurred during the load and the number of rows that were skipped.

- **MESSAGE LOG** – specifies the names of files in which to log information about integrity constraint violations and the types of violations to log. Timestamps indicating the start and completion of the load are logged in both the `MESSAGE LOG` and the `ROW LOG` files. Both `MESSAGE LOG` and `ROW LOG` must be specified, or no information about integrity violations is logged.
 - If the `ONLY LOG` clause is not specified, no information on integrity constraint violations is logged. Only the timestamps indicating the start and completion of the load are logged.
 - Information is logged on all integrity constraint-type violations specified in the `ONLY LOG` clause or for all word index-length violations if the keyword `WORD` is specified.
 - If constraint violations are being logged, every occurrence of an integrity constraint violation generates exactly one row of information in the `MESSAGE LOG` file. The number of rows (errors reported) in the `MESSAGE LOG` file can exceed the `IGNORE CONSTRAINT` option limit, because the load is performed by multiple threads running in parallel. More than one thread might report that the number of constraint violations has exceeded the specified limit.
 - If constraint violations are being logged, exactly one row of information is logged in the `ROW LOG` file for a given row, regardless of the number of integrity constraint violations that occur on that row.

The number of distinct errors in the `MESSAGE LOG` file might not exactly match the number of rows in the `ROW LOG` file. The difference in the number of rows is due to the parallel processing of the load described above for the `MESSAGE LOG`.

- The MESSAGE LOG and ROW LOG files cannot be raw partitions or named pipes.
- If the MESSAGE LOG or ROW LOG file already exists, new information is appended to the file.
- Specifying an invalid file name for the MESSAGE LOG or ROW LOG file generates an error.
- Specifying the same file name for the MESSAGE LOG and ROW LOG files generates an error.

Various combinations of the IGNORE CONSTRAINT and MESSAGE LOG options result in different logging actions.

Table 5. LOAD TABLE Logging Actions

IGNORE CONSTRAINT Specified?	MESSAGE LOG Specified?	Action
yes	yes	All ignored integrity constraint violations are logged, including the user specified limit, before the rollback.
no	yes	The first integrity constraint violation is logged before the rollback.
yes	no	Nothing is logged.
no	no	Nothing is logged. The first integrity constraint violation causes a rollback.

Tip: Set the IGNORE CONSTRAINT option limit to a nonzero value, if you are logging the ignored integrity constraint violations. If a single row has more than one integrity constraint violation, a row for each violation is written to the MESSAGE LOG file. Logging an excessive number of violations affects the performance of the load.

- **LOG DELIMITED BY** – specifies the separator between data values in the ROW LOG file. The default separator is a comma.

SAP Sybase IQ no longer returns an error message when FORMAT BCP is specified as a **LOAD TABLE** clause. In addition, these conditions are verified and proper error messages are returned

- If the specified load format is not ASCII, BINARY, or BCP, SAP Sybase IQ returns the message “Only ASCII, BCP and BINARY are supported LOAD formats.”
- If the **LOAD TABLE** column specification contains anything other than column name, NULL, or ENCRYPTED, then SAP Sybase IQ returns the error message “Invalid load specification for LOAD ... FORMAT BCP.”

- If the column delimiter or row terminator size for the FORMAT BCP load is greater than 10 characters, then SAP Sybase IQ returns the message “Delimiter ‘%2’ must be 1 to %3 characters in length.” (where %3 equals 10). Messages corresponding to error or warning conditions which can occur for FORMAT BCP as well as FORMAT ASCII are the same for both formats.
- If the load default value specified is AUTOINCREMENT, IDENTITY, or GLOBAL AUTOINCREMENT, SAP Sybase IQ returns the error “Default value %2 cannot be used as a LOAD default value. %1”
- If the **LOAD TABLE** specification does not contain any columns that need to be loaded from the file specified, SAP Sybase IQ returns the error “The LOAD statement must contain at least one column to be loaded from input file.” and the **LOAD TABLE** statement rolls back.
- If a load exceeds the limit on the maximum number of terms for a text document with TEXT indexes, SAP Sybase IQ returns the error “Text document exceeds maximum number of terms. Support up to 4294967295 terms per document.”

Examples

- **Example 1** – load data from one file into the Products table on a Windows system. A tab is used as the column delimiter following the Description and Color columns:

```
LOAD TABLE Products
( ID ASCII(6),
  FILLER(1),
  Name ASCII(15),
  FILLER(1),
  Description '\x09',
  Size ASCII(2),
  FILLER(1),
  Color '\x09',
  Quantity PREFIX 2,
  UnitPrice PREFIX 2,
  FILLER(2) )
FROM 'C:\mydata\source1.dmp'
QUOTES OFF
ESCAPES OFF
BYTE ORDER LOW
NOTIFY 1000
```

- **Example 2** – load data from a file a.inp on a client computer:

```
LOAD TABLE t1(c1,c2,filler(30))
USING CLIENT FILE 'c:\client-data\a.inp'
QUOTES OFF ESCAPES OFF
IGNORE CONSTRAINT UNIQUE 0, NULL 0
MESSAGE LOG 'c:\client-data\m.log'
ROW LOG 'c:\client-data\r.log'ONLY LOG UNIQUE
```

SQL Statements

- **Example 3** – load data from two files into the `product_new` table (which allows NULL values) on a UNIX system. The tab character is the default column delimiter, and the newline character is the row delimiter:

```
LOAD TABLE product_new
( id,
  name,
  description,
  size,
  color   '\x09'   NULL( 'null', 'none', 'na' ),
  quantity PREFIX 2,
  unit_price PREFIX 2 )
FROM '/s1/mydata/source2.dump',
     '/s1/mydata/source3.dump'
QUOTES OFF
ESCAPES OFF
FORMAT ascii
DELIMITED BY '\x09'
ON FILE ERROR CONTINUE
ROW DELIMITED BY '\n'
```

- **Example 4** – ignore 10 word-length violations; on the 11th, deploy the new error and roll back the load:

```
load table PTAB1(
  ck1      ',,' null ('NULL') ,
  ck3fk2c2 ',,' null ('NULL') ,
  ck4      ',,' null ('NULL') ,
  ck5      ',,' null ('NULL') ,
  ck6c1    ',,' null ('NULL') ,
  ck6c2    ',,' null ('NULL') ,
  rid      ',,' null ('NULL') )
FROM 'ri_index_selfRI.inp'
row delimited by '\n'
LIMIT 14 SKIP 10
IGNORE CONSTRAINT UNIQUE 2, FOREIGN KEY 8
word skip 10 quotes off escapes off strip
off
```

- **Example 5** – load data into table `t1` from the **BCP** character file `bcp_file.bcp` using the **FORMAT BCP** load option:

```
LOAD TABLE t1 (c1, c2, c3)
FROM 'bcp_file.bcp'
FORMAT BCP
...
```

- **Example 6** – load default values 12345 into `c1` using the **DEFAULT** load option, and load `c2` and `c3` with data from the `LoadConst04.dat` file:

```
LOAD TABLE t1 (c1 DEFAULT '12345 ', c2, c3, filler(1))
FROM 'LoadConst04.dat'
STRIP OFF
QUOTES OFF
```

```
ESCAPES OFF
DELIMITED BY ',';
```

- **Example 7** – load `c1` and `c2` with data from the file `bcp_file.bcp` using the **FORMAT BCP** load option and set `c3` to the value 10:

```
LOAD TABLE t1 (c1, c2, c3 DEFAULT '10')
FROM 'bcp_file.bcp'
FORMAT BCP
QUOTES OFF
ESCAPES OFF;
```

- **Example 8** – this code fragment ignores one header row at the beginning of the data file, where the header row is delimited by '&&':

```
LOAD TABLE
...HEADER SKIP 1 HEADER DELIMITED BY '&&'
```

- **Example 9** – this code fragment ignores 2 header rows at the beginning of the data file, where each header row is delimited by '\n':

```
LOAD TABLE
...HEADER SKIP 2
```

- **Example 10** – load a file into a RLV-enabled table.

Load data into RLV-enabled table `rvt1` from the **BCP** character file `bcp_file.bcp` using the **FORMAT BCP** load option:

```
LOAD TABLE rvt1 (c1, c2, c3)
FROM 'bcp_file.bcp'
FORMAT BCP
...
```

Usage

The **LOAD TABLE** statement allows efficient mass insertion into a database table from a file with ASCII or binary data.

The **LOAD TABLE** options also let you control load behavior when integrity constraints are violated and to log information about the violations.

You can use **LOAD TABLE** on a temporary table, but the temporary table must have been declared with **ON COMMIT PRESERVE ROWS**, or the next **COMMIT** removes the rows you have loaded.

LOAD TABLE supports loading of large object (LOB) data.

SAP Sybase IQ supports loading from both ASCII and binary data, and it supports both fixed- and variable-length formats. To handle all of these formats, you must supply a *load-specification* to tell SAP Sybase IQ what kind of data to expect from each “column” or field in the source file. The *column-spec* lets you define these formats:

SQL Statements

- ASCII with a fixed length of bytes. The *input-width* value is an integer indicating the fixed width in bytes of the input field in every record.
- Binary or non-binary fields that use a number of PREFIX bytes (1, 2, or 4) to specify the length of the input.

There are two parts related to a **PREFIX** clause:

- Prefix value – always a binary value.
- Associated data bytes – always character format; never binary format.

If the data is unloaded using the extraction facility with the `TEMP_EXTRACT_BINARY` option set ON, you must use the **BINARY WITH NULL BYTE** parameter for each column when you load the binary data.

- Variable-length characters delimited by a separator. You can specify the terminator as hexadecimal ASCII characters. The *delimiter-string* can be any string of up to 4 characters, including any combination of printable characters, and any 8-bit hexadecimal ASCII code that represents a nonprinting character. For example, specify:
 - '\x09' to represent a tab as the terminator.
 - '\x00' for a null terminator (no visible terminator as in “C” strings).
 - '\x0a' for a newline character as the terminator. You can also use the special character combination of '\n' for newline.

Note: The delimiter string can be from 1 to 4 characters long, but you can specify only a single character in the **DELIMITED BY** clause. For **BCP**, the delimiter can be up to 10 characters.

- DATE or DATETIME string as ASCII characters. You must define the *input-date-format* or *input-datetime-format* of the string using one of the corresponding formats for the date and datetime data types supported by SAP Sybase IQ. Use **DATE** for date values and **DATETIME** for datetime and time values.

Table 6. Formatting Dates and Times

Option	Meaning
yyyy or YYYY yy or YY	Represents number of year. Default is current year.
mm or MM	Represents number of month. Always use leading zero or blank for number of the month where appropriate, for example, '05' for May. DATE value must include a month. For example, if the DATE value you enter is 1998, you receive an error. If you enter '03', SAP Sybase IQ applies the default year and day and converts it to '1998-03-01'.
dd or DD jjj or JJJ	Represents number of day. Default day is 01. Always use leading zeros for number of day where appropriate, for example, '01' for first day. J or j indicates a Julian day (1 to 366) of the year.

Option	Meaning
hh HH	Represents hour. Hour is based on 24-hour clock. Always use leading zeros or blanks for hour where appropriate, for example, '01' for 1 am. '00' is also valid value for hour of 12 a.m.
nn	Represents minute. Always use leading zeros for minute where appropriate, for example, '08' for 8 minutes.
ss[.sssss]	Represents seconds and fraction of a second.
aa	Represents the a.m. or p.m. designation.
pp	Represents the p.m. designation only if needed. (This is an incompatibility with SAP Sybase IQ versions earlier than 12.0; previously, "pp" was synonymous with "aa".)
hh	SAP Sybase IQ assumes zero for minutes and seconds. For example, if the DATETIME value you enter is '03', SAP Sybase IQ converts it to '03:00:00.0000'.
hh:nn or hh:mm	SAP Sybase IQ assumes zero for seconds. For example, if the time value you enter is '03:25', SAP Sybase IQ converts it to '03:25:00.0000'.

Table 7. Sample DATE and DATETIME Format Options

Input data	Format specification
12/31/98	DATE ('MM/DD/YY')
19981231	DATE ('YYYYMMDD')
123198140150	DATETIME ('MMDDYYhhnnss')
14:01:50 12-31-98	DATETIME ('hh:nn:ss MM-DD-YY')
18:27:53	DATETIME ('hh:nn:ss')
12/31/98 02:01:50AM	DATETIME ('MM/DD/YY hh:nn:ssaa')

SAP Sybase IQ has built-in load optimizations for common date, time, and datetime formats. If your data to be loaded matches one of these formats, you can significantly decrease load time by using the appropriate format.

You can also specify the date/time field as an ASCII fixed-width field (as described above) and use the **FILLER(1)** option to skip the column delimiter.

The **NULL** portion of the *column-spec* indicates how to treat certain input values as **NULL** values when loading into the table column. These characters can include **BLANKS**, **ZEROS**, or any other list of literals you define. When specifying a **NULL** value or reading a **NULL** value from the source file, the destination column must be able to contain **NULLs**.

ZEROS are interpreted as follows: the cell is set to **NULL** if (and only if) the input data (before conversion, if ASCII) is all binary zeros (and not character zeros).

- If the input data is character zero, then:
 1. NULL (ZEROS) never causes the cell to be NULL.
 2. NULL ('0') causes the cell to be NULL.
- If the input data is binary zero (all bits clear), then:
 1. NULL (ZEROS) causes the cell to be NULL.
 2. NULL ('0') never causes the cell to be NULL.

For example, if your **LOAD** statement includes `coll date('yymmdd') null(zeros)` and the date is 000000, you receive an error indicating that 000000 cannot be converted to a `DATE(4)`. To get **LOAD TABLE** to insert a NULL value in `coll` when the data is 000000, either write the NULL clause as `null('000000')`, or modify the data to equal binary zeros and use `NULL(ZEROS)`.

If the length of a `VARCHAR` cell is zero and the cell is not NULL, you get a zero-length cell. For all other data types, if the length of the cell is zero, SAP Sybase IQ inserts a NULL. This is ANSI behavior. For non-ANSI treatment of zero-length character data, set the **NON_ANSI_NULL_VARCHAR** database option.

Use the **DEFAULT** option to specify a load default column value. You can load a default value into a column, even if the column does not have a default value defined in the table schema. This feature provides more flexibility at load time.

- The **LOAD TABLE DEFAULTS** option must be ON in order to use the default value specified in the **LOAD TABLE** statement. If the **DEFAULTS** option is OFF, the specified load default value is not used and a NULL value is inserted into the column instead.
- The **LOAD TABLE** command must contain at least one column that needs to be loaded from the file specified in the **LOAD TABLE** command. Otherwise, an error is reported and the load is not performed.
- The specified load default value must conform to the supported default values for columns and default value restrictions. The **LOAD TABLE DEFAULT** option does not support **AUTOINCREMENT**, **IDENTITY**, or **GLOBAL AUTOINCREMENT** as a load default value.
- The **LOAD TABLE DEFAULT** *default-value* must be of the same character set as that of the database.
- Encryption of the default value is not supported for the load default values specified in the **LOAD TABLE DEFAULT** clause.
- A constraint violation caused by evaluation of the specified load default value is counted for each row that is inserted in the table.

Another important part of the *load-specification* is the **FILLER** option. This option indicates you want to skip over a specified field in the source input file. For example, there may be characters at the end of rows or even entire fields in the input files that you do not want to add to the table. As with the *column-spec* definition, **FILLER** specifies ASCII fixed length of bytes, variable length characters delimited by a separator, and binary fields using **PREFIX** bytes.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

The permissions required to execute a **LOAD TABLE** statement depend on the database server **-gl** command line option, as follows:

- **-gl ALL** – You must be the owner of the table, have ALTER or LOAD permission on the table, or have the ALTER ANY TABLE, LOAD ANY TABLE, or ALTER ANY OBJECT system privilege.
- **-gl DBA** – You must have the ALTER ANY TABLE, LOAD ANY TABLE, or ALTER ANY OBJECT system privilege.
- **-gl NONE** – Execution of the **LOAD TABLE** statement is not permitted.

For more information on the **-gl** command line option, please refer *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options*.

LOAD TABLE also requires a write lock on the table.

See also

- *INSERT Statement* on page 327
- *LOAD_ZEROLENGTH_ASNULL Option* on page 568
- *NON_ANSI_NULL_VARCHAR Option* on page 593

Storage Sizes

The storage size of character data, given column definition size and input data size.

Table 8. Storage Size of Character Data

Data type	Column definition	Input data	Storage
CHARACTER, CHAR	width of (32K – 1) bytes	(32K – 1) bytes	(32K – 1) bytes
VARCHAR, CHARACTER VARYING	width of (32K – 1) bytes	(32K – 1) bytes	(32K – 1) bytes

LOCK TABLE Statement

Prevents other concurrent transactions from accessing or modifying a table within the specified time.

Syntax

```
LOCK TABLE table-list [ WITH HOLD ]
      IN { SHARE | WRITE | EXCLUSIVE } MODE [ WAIT time ]
```

```
table-list:
  [ owner. ] table-name [ , [ owner. ] table-name, ... ]
```

Parameters

- **table-name** – must be a base table, not a view. WRITE mode is only valid for IQ base tables. **LOCK TABLE** either locks all tables in the table list, or none. The table must not be enabled for row-level versioning (RLV). If obtaining a lock for a SQL Anywhere table, or when obtaining SHARE or EXCLUSIVE locks, you may only specify a single table. Standard SAP Sybase IQ object qualification rules are used to parse *table-name*.
- **WITH HOLD** – the lock is held until the end of the connection. If the clause is not specified, the lock is released when the current transaction is committed or rolled back. Using the WITH HOLD clause in the same statement with WRITE MODE is unsupported and returns the error SQLCODE=-131, ODBC 3 State="42000".
- **SHARE** – prevents other transactions from modifying the table, but allows them read access. In this mode, you can change data in the table as long as no other transaction has locked the row being modified, either indirectly, or explicitly by using **LOCK TABLE**.
- **WRITE** – prevents other transactions from modifying a list of tables. Unconditionally commits the connections outermost transaction. The transaction's snapshot version is established not by the **LOCK TABLE IN WRITE MODE** statement, but by the execution of the next command processed by SAP Sybase IQ.

WRITE mode locks are released when the transaction commits or rolls back, or when the connection disconnects.

- **EXCLUSIVE** – prevents other transactions from accessing the table. In this mode, no other transaction can execute queries, updates of any kind, or any other action against the table.
- **WAIT time** – specifies maximum blocking time for all lock types. This clause is mandatory when lock mode is WRITE. When a time argument is given, the server locks the specified tables only if available within the specified time. The time argument can be specified in the format hh:nn:ss:sss. If a date part is specified, the server ignores it and

converts the argument into a timestamp. When no time argument is given, the server waits indefinitely until a **WRITE** lock is available or an interrupt occurs.

Examples

- **Example 1** – obtain a **WRITE** lock on the `Customers` and `Employees` tables, if available within 5 minutes and 3 seconds:

```
LOCK TABLE Customers, Employees IN WRITE MODE WAIT
'00:05:03'
```

- **Example 2** – wait indefinitely until the **WRITE** lock on the `Customers` and `Employees` tables is available, or an interrupt occurs:

```
LOCK TABLE Customers, Employees IN WRITE MODE WAIT
```

Usage

LOCK TABLE statements run on tables in the IQ main store on the coordinator do not affect access to those tables from connections on secondary servers. For example:

On a coordinator connection, issue the command:

```
LOCK TABLE coord1 WITH HOLD IN EXCLUSIVE MODE
```

sp_iqlocks on the coordinator confirms that the table `coord1` has an exclusive (E) lock.

The result of **sp_iqlocks** run on a connection on a secondary server does not show the exclusive lock on table `coord1`. The user on this connection can see updates to table `coord1` on the coordinator.

Other connections on the coordinator can see the exclusive lock on `coord1` and attempting to select from table `coord1` from another connection on the coordinator returns `User DBA has the row in coord1 locked.`

LOCK TABLE on views is unsupported. Attempting to lock a view acquires a shared schema lock regardless of the mode specified in the command. A shared schema lock prevents other transactions from modifying the table schema.

The Transact-SQL (T-SQL) stored procedure dialect does not support **LOCK TABLE**. For example, this statement returns `Syntax error near LOCK:`

```
CREATE PROCEDURE tproc()
AS
BEGIN
COMMIT;
LOCK TABLE t1 IN SHARE MODE
INSERT INTO t1 VALUES(30)
END
```

The Watcom-SQL stored procedure dialect supports **LOCK TABLE**. The default command delimiter is a semicolon (;). For example:

SQL Statements

```
CREATE PROCEDURE tproc ()
AS
BEGIN
COMMIT;
LOCK TABLE t1 IN SHARE MODE
INSERT INTO t1 VALUES (30)
END
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- SAP Sybase—Supported in Adaptive Server Enterprise. The **WITH HOLD** clause is not supported in Adaptive Server Enterprise. Adaptive Server Enterprise provides a **WAIT** clause that is not supported in SQL Anywhere.

Permissions

To lock a table in SHARE mode, SELECT privileges are required.

To lock a table in EXCLUSIVE mode, you must be the table owner or have any of the following system privileges:

- ALTER ANY OBJECT
- INSERT ANY TABLE
- UPDATE ANY TABLE
- DELETE ANY TABLE
- ALTER ANY TABLE
- LOAD ANY TABLE
- TRUNCATE ANY TABLE

.

See also

- *SELECT Statement* on page 417

LOOP Statement

Repeats the execution of a statement list.

Syntax

```
[ statement-label: ]
... [ WHILE search-condition ] LOOP
... statement-list
... END LOOP [ statement-label ]
```

Examples

- **Example 1** – a **WHILE** loop in a procedure:

```
...
SET i = 1 ;
WHILE i <= 10 LOOP
    INSERT INTO Counters( number ) VALUES ( i ) ;
    SET i = i + 1 ;
END LOOP ;
...
```

- **Example 2** – a labeled loop in a procedure:

```
SET i = 1;
lbl:
LOOP
    INSERT
    INTO Counters( number )
    VALUES ( i ) ;
    IF i >= 10 THEN
        LEAVE lbl ;
    END IF ;
    SET i = i + 1 ;
END LOOP lbl
```

Usage

The **WHILE** and **LOOP** statements are control statements that let you repeatedly execute a list of SQL statements while a *search-condition* evaluates to TRUE. The **LEAVE** statement can be used to resume execution at the first statement after the **END LOOP**.

If the ending *statement-label* is specified, it must match the beginning *statement-label*.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported in Adaptive Server Enterprise. The **WHILE** statement provides looping in Transact-SQL stored procedures.

Permissions

None

See also

- *FOR Statement* on page 286
- *LEAVE Statement* on page 342
- *WHILE Statement [T-SQL]* on page 460

MESSAGE Statement

Displays a message, which can be any expression. Clauses can specify where the message is displayed.

Syntax

```
MESSAGE expression, ...
  [ TYPE { INFO | ACTION | WARNING | STATUS } ]
  [ TO { CONSOLE
    | CLIENT [ FOR { CONNECTION conn_id [ IMMEDIATE ] | ALL } ]
    | [ EVENT | SYSTEM ] LOG }
  [ DEBUG ONLY ] ]
```

Parameters

- **FOR** – specifies which connections receive notification about the message:
 - **CONNECTION *conn_id*** – the recipient's connection ID for the message.
 - **IMMEDIATE** – the connection receives the message within a few seconds regardless of when the SQL statement is executed.

Typically, messages sent using the IMMEDIATE clause are delivered in less than five seconds, even if the destination connection is not making database server requests.

Message delivery could be delayed if the client connection makes several requests per second, receives very large BLOB data, or if the client's message callback executes for more than a second. In addition, sending more than one IMMEDIATE message to a single connection every two seconds could delay message delivery or generate an error message. If the client connection is disconnected, a successful MESSAGE...IMMEDIATE statement may not be delivered.

- **ALL** – all open connections receive the message.

The FOR clause can be used to notify another application of an event detected on the server without the need for the application to explicitly check for the event. When the FOR clause is used, recipients receive the message the next time they execute a SQL statement. If the recipient is currently executing a SQL statement, the message is received when the statement completes. If the statement being executed is a stored procedure call, the message is received before the call is completed.

If an application requires notification within a short time after the message is sent and when the connection is not executing SQL statements, you can use a second connection. This connection can execute one or more **WAITFOR DELAY** statements. These statements do not consume significant resources on the server or network (as would happen with a polling approach), but permit applications to receive notification of the message shortly after it is sent.

- **TYPE** – has an effect only if the message is sent to the client. The client application must decide how to handle the message. Interactive SQL displays messages in these locations:
 - **INFO** – (default) The Message window.
 - **ACTION** – a Message box with an OK button.
 - **WARNING** – a Message box with an OK button.
 - **STATUS** – the Messages pane.
- **TO** – specifies the destination of a message:
 - **CONSOLE** – (default) Send messages to the database server window.
 - **CLIENT** – send messages to the client application. Your application must decide how to handle the message, and you can use the **TYPE** clause as information on which to base that decision.
 - **LOG** – send messages to the server log file specified by the **-o** option.
- **DEBUG ONLY** – controls whether debugging messages added to stored procedures are enabled or disabled by changing the setting of the **DEBUG_MESSAGES** database option. When **DEBUG ONLY** is specified, the **MESSAGE** statement is executed only when the **DEBUG_MESSAGES** option is set to **ON**.

Note: **DEBUG ONLY** messages are inexpensive when the **DEBUG_MESSAGES** option is set to **OFF**, so these statements can usually be left in stored procedures on a production system. However, they should be used sparingly in locations where they would be executed frequently; otherwise, they might result in a small performance penalty.

Examples

- **Example 1** – display the string `The current date and time`, and the current date and time, on the database server message window:

```
CREATE PROCEDURE message_test ()
BEGIN
MESSAGE 'The current date and time: ', Now();
END;
CALL message_test();
```

- **Example 2** – to register a callback in ODBC, first declare the message handler:

```
void SQL_CALLBACK my_msgproc (
    void *      sqlca,
    unsigned char  msg_type,
    long        code,
    unsigned short len,
    char*       msg )
{ ... }
```

Install the declared message handler by calling the `SQLSetConnectAttr` function:

```
rc = SQLSetConnectAttr(
    dbc,
```

SQL Statements

```
ASA_REGISTER_MESSAGE_CALLBACK,  
(SQLPOINTER) &my_msgproc, SQL_IS_POINTER );
```

Usage

The procedure issuing a **MESSAGE ... TO CLIENT** statement must be associated with a connection.

For example, the message box is not displayed because the event occurs outside of a connection:

```
CREATE EVENT CheckIdleTime TYPE ServerIdle  
WHERE event_condition( 'IdleTime' ) > 100  
HANDLER  
BEGIN  
    MESSAGE 'Idle engine' type warning to client;  
END;
```

However, in this example, the message is written to the server console:

```
CREATE EVENT CheckIdleTime TYPE ServerIdle  
WHERE event_condition( 'IdleTime' ) > 100  
HANDLER  
BEGIN  
    MESSAGE 'Idle engine' type warning to console;  
END;
```

Valid expressions can include a quoted string or other constant, variable, or function. However, queries are not permitted in the output of a **MESSAGE** statement, even though the definition of an expression includes queries.

ESQL and ODBC clients receive messages via message callback functions. In each case, these functions must be registered. To register ESQL message handlers, use the **db_register_callback** function.

ODBC clients can register callback functions using the **SQLSetConnectAttr** function.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported in Adaptive Server Enterprise. The Transact-SQL **PRINT** statement provides a similar feature, and is available in SQL Anywhere.

Permissions

Must be connected to the database.

- **FOR** clause – Requires one of:
 - **SERVER OPERATOR** system privilege.
 - **DROP CONNECTION** system privilege.

- **TO EVENT LOG** or **TO SYSTEM LOG** clause – Requires the **SERVER OPERATOR** system privilege.

See also

- *CREATE PROCEDURE Statement* on page 165
- *WAITFOR Statement* on page 458
- *DEBUG_MESSAGES Option* on page 516

OPEN Statement [ESQL] [SP]

Opens a previously declared cursor to access information from the database.

Syntax

```
OPEN cursor-name
... [ USING [ DESCRIPTOR { sqlda-name | host-variable [, ...] } ] ]
... [ WITH HOLD ]
```

Parameters

- **cursor-name** – identifier or host-variable.
If the cursor name is specified by an identifier or string, then the corresponding **DECLARE CURSOR** statement must appear prior to the **OPEN** in the C program; if the cursor name is specified by a host variable, then the **DECLARE CURSOR** statement must execute before the **OPEN** statement.
- **sqlda-name** – identifier
- **WITH HOLD** – keeps the cursor open for subsequent transactions. The cursor remains open until the end of the current connection or until an explicit **CLOSE** statement is executed. Cursors are automatically closed when a connection is terminated.
- **USING** – specifies the host variables that are bound to the placeholder bind variables in the **SELECT** statement for which the cursor has been declared.

Examples

- **Example 1** – use of **OPEN** in Embedded SQL:

```
EXEC SQL OPEN employee_cursor;
```

and

```
EXEC SQL PREPARE emp_stat FROM
'SELECT EmployeeID, Surname FROM Employees WHERE name like ?';
EXEC SQL DECLARE employee_cursor CURSOR FOR emp_stat;
EXEC SQL OPEN employee_cursor USING :pattern;
```

SQL Statements

- **Example 2** – an example from a procedure:

```
BEGIN
DECLARE cur_employee CURSOR FOR
  SELECT Surname
  FROM Employees ;
DECLARE name CHAR(40) ;
OPEN cur_employee;
LOOP
FETCH NEXT cur_employee into name ;
  ...
END LOOP
CLOSE cur_employee;
END
```

Usage

By default, all cursors are automatically closed at the end of the current transaction (**COMMIT** or **ROLLBACK**).

The cursor is positioned before the first row.

A cursor declared using the **FOR READ ONLY** clause sees the version of table(s) on which the cursor is declared when the cursor is opened, not the version of table(s) at the time of the first **FETCH** statement.

The **USING DESCRIPTOR** *sqlda-name*, **host-variable**, and **BLOCK n** clauses are for Embedded SQL only.

After successful execution of the **OPEN** statement, the `sqlerrd[3]` field of the **SQLCA** (**SQLIOESTIMATE**) is filled in with an estimate of the number of input/output operations required to fetch all rows of the query. Also, the `sqlerrd[2]` field of the **SQLCA** (**SQLCOUNT**) is filled in with either the actual number of rows in the cursor (a value greater than or equal to 0), or an estimate thereof (a negative number whose absolute value is the estimate). The `sqlerrd[2]` field is the actual number of rows, if the database server can compute this value without counting the rows.

Standards

- **SQL**—ISO/ANSI SQL compliant.
- **Sybase**—The simple **OPEN** *cursor-name* syntax is supported by Adaptive Server Enterprise. None of the other clauses are supported in Adaptive Server Enterprise stored procedures. Open Client/Open Server supports the **USING** descriptor or host name variable syntax.

Permissions

- Must have **SELECT** permission on all tables in a **SELECT** statement or **EXECUTE** permission on the procedure in a **CALL** statement.

- When the cursor is on a **CALL** statement, **OPEN** causes the procedure to execute until the first result set (**SELECT** statement with no **INTO** clause) is encountered. If the procedure completes and no result set is found, the `SQLSTATE_PROCEDURE_COMPLETE` warning is set.

See also

- *CLOSE Statement [ESQL] [SP]* on page 95
- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *FETCH Statement [ESQL] [SP]* on page 283
- *PREPARE Statement [ESQL]* on page 375
- *RESUME Statement* on page 395

OUTPUT Statement [Interactive SQL]

Writes the information retrieved by the current query to a file.

Syntax

```
OUTPUT TO filename
  [ APPEND ] [ VERBOSE ]
  [ FORMAT output-format ]
  [ ESCAPE CHARACTER character ]
  [ DELIMITED BY string ]
  [ QUOTE string [ ALL ] ]
  [ COLUMN WIDTHS ( integer, ... ) ]
  [ HEXADEDECIMAL { ON | OFF | ASIS } ]
  [ ENCODING encoding ]
  [ WITH COLUMN NAMES ]
```

output-format:
TEXT | **FIXED** | **HTML** | **SQL** | **XML**

Parameters

- **FORMAT** – the output format. If no **FORMAT** clause is specified, the Interactive SQL `OUTPUT_FORMAT` database option setting is used.
- **TEXT** – output is a **TEXT** format file with one row per line in the file. All values are separated by commas, and strings are enclosed in apostrophes (single quotes). The delimiter and quote strings can be changed using the **DELIMITED BY** and **QUOTE** clauses. If the **ALL** clause is specified in the **QUOTE** clause, all values (not just strings) are quoted. **TEXT** is the default output format.

Three other special sequences are also used. The two characters `\n` represent a newline character, `\\` represents a single `\`, and the sequence `\xDD` represents the character with hexadecimal code `DD`.

If you are exporting Java methods that have string return values, you must use the **HEXADECIMAL OFF** clause.

- **FIXED** – output is fixed format with each column having a fixed width. The width for each column can be specified using the **COLUMN WIDTHS** clause. No column headings are output in this format.

If the **COLUMN WIDTHS** clause is omitted, the width for each column is computed from the data type for the column, and is large enough to hold any value of that data type. The exception is that **LONG VARCHAR** and **LONG BINARY** data defaults to 32KB.

- **HTML** – output is in the Hyper Text Markup Language format.
- **SQL** – output is an Interactive SQL **INPUT** statement required to recreate the information in the table.

Note: SAP Sybase IQ does not support the **INPUT** statement. You need to edit this statement to a valid **LOAD TABLE** (or **INSERT**) statement to use it to load data back in.

- **XML** – output is an XML file encoded in UTF-8 and containing an embedded DTD. Binary values are encoded in CDATA blocks with the binary data rendered as 2-hex-digit strings. The **LOAD TABLE** statement does not accept XML as a file format.
- **APPEND** – appends the results of the query to the end of an existing output file without overwriting the previous contents of the file. If the **APPEND** clause is not used, the **OUTPUT** statement overwrites the contents of the output file by default.

The **APPEND** clause is valid if the output format is **TEXT**, **FIXED**, or **SQL**.

- **VERBOSE** – error messages about the query, the SQL statement used to select the data, and the data itself are written to the output file. If the **VERBOSE** clause is omitted (the default), only the data is written to the file. The **VERBOSE** clause is valid if the output format is **TEXT**, **FIXED**, or **SQL**.
- **ESCAPE CHARACTER** – the default escape character for characters stored as hexadecimal codes and symbols is a backslash (\), so `\x0A` is the line feed character, for example.

This default can be changed using the **ESCAPE CHARACTER** clause. For example, to use the exclamation mark as the escape character, enter:

```
... ESCAPE CHARACTER '!'
```

- **DELIMITED BY** – for the **TEXT** output format only. The delimiter string is placed between columns (default comma).
- **QUOTE** – for the **TEXT** output format only. The quote string is placed around string values. The default is a single quote character. If **ALL** is specified in the **QUOTE** clause, the quote string is placed around all values, not just around strings.
- **COLUMN WIDTHS** – specifies the column widths for the **FIXED** format output.

- **HEXADECIMAL** – specifies how binary data is to be unloaded for the TEXT format only. When set to ON, binary data is unloaded in the format 0xabcd. When set to OFF, binary data is escaped when unloaded (\xab\xcd). When set to ASIS, values are written as is, that is, without any escaping—even if the value contains control characters. ASIS is useful for text that contains formatting characters such as tabs or carriage returns.
- **ENCODING** – specifies the encoding that is used to write the file. The ENCODING clause can be used only with the TEXT format. Can be a string or identifier.

If the ENCODING clause is not specified, Interactive SQL determines the code page that is used to write the file as follows, where code page values occurring earlier in the list take precedence over those occurring later:

- The code page specified with the **DEFAULT_ISQL_ENCODING** option (if this option is set)
- The default code page for the computer Interactive SQL is running on

Examples

- **Example 1** – places the contents of the `Employees` table in a text file:

```
SELECT * FROM Employees;
OUTPUT TO employees.txt FORMAT TEXT
```

- **Example 2** – places the contents of the `Employees` table at the end of an existing file, and include any messages about the query in this file as well:

```
SELECT * FROM Employees;
OUTPUT TO employees.txt APPEND VERBOSE
```

- **Example 3** – exports a value that contains an embedded line feed character. A line feed character has the numeric value 10, which you can represent as the string `\x0a` in a SQL statement.

Execute this statement with **HEXADECIMAL ON**:

```
SELECT 'line1\x0aline2'; OUTPUT TO file.txt HEXADECIMAL ON
```

The result is a file with one line in it, containing this text:

```
line10x0aline2
```

Execute the same statement with **HEXADECIMAL OFF**:

```
line1\x0aline2
```

If you set **HEXADECIMAL** to **ASIS**, you get a file with two lines:

```
'line1
line2'
```

Using **ASIS** generates two lines, because the embedded line feed character has been exported without being converted to a two-digit hex representation, and without a prefix.

Usage

The current query is the **SELECT** or **LOAD TABLE** statement that generated the information that appears on the Results tab in the Results pane. The **OUTPUT** statement reports an error if there is no current query.

Note: **OUTPUT** is especially useful in making the results of a query or report available to another application, but is not recommended for bulk operations. For high-volume data movement, use the ASCII and BINARY data extraction functionality with the **SELECT** statement. The extraction functionality provides much better performance for large-scale data movement, and creates an output file you can use for loads.

Side Effects

- In Interactive SQL, the Results tab displays only the results of the current query. All previous query results are replaced with the current query results.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *SELECT Statement* on page 417
- *DEFAULT_ISQL_ENCODING Option [Interactive SQL]* on page 520

PARAMETERS Statement [Interactive SQL]

Specifies parameters to an Interactive SQL (**dbisql**) command file.

Syntax

```
PARAMETERS parameter1, parameter2, ...
```

Examples

- **Example 1** – this **dbisql** command file takes two parameters:

```
PARAMETERS department_id, file ;
SELECT Surname
FROM Employees
WHERE DepartmentID = {department_id}
>#{file}.dat;
```


Usage

PARAMETERS specifies how many parameters there are to a command file and also names those parameters so that they can be referenced later in the command file.

Parameters are referenced by putting the named parameter into the command file where you want the parameter to be substituted:

```
{parameter1}
```

There must be no spaces between the braces and the parameter name.

If a command file is invoked with fewer than the required number of parameters, **dbisql** prompts for values of the missing parameters.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *READ Statement [Interactive SQL]* on page 382

PREPARE Statement [ESQL]

Prepares a statement to be executed later or used for a cursor.

Syntax

```
PREPARE statement-name
FROM statement [ FOR { READ ONLY | UPDATE [ OF column-name-list ] } ]
... [ DESCRIBE describe-type INTO [ [ SQL ] DESCRIPTOR ] descriptor ]
... [ WITH EXECUTE ]
```

describe-type:

```
{ ALL
| BIND VARIABLES
| INPUT
| OUTPUT
| SELECT LIST } ... { LONG NAMES [ [ OWNER.]TABLE.]COLUMN ]
| WITH VARIABLE RESULT }
```

Parameters

- **statement-name** – referenced to execute the statement, or to open a cursor if the statement is a **SELECT** statement. *statement-name* may be a host variable of type `a_sql_statement_number` defined in the `sqlca.h` header file that is automatically included. If an identifier is used for the *statement-name*, only one statement per module may be prepared with this *statement-name*.
- **FOR UPDATE | FOR READ ONLY** – defines the cursor updatability if the statement is used by a cursor. A FOR READ ONLY cursor cannot be used in an **UPDATE** (positioned) or a **DELETE** (positioned) operation. FOR READ ONLY is the default. In response to any request for a cursor that specifies FOR UPDATE, SAP Sybase IQ provides either a value-sensitive cursor or a sensitive cursor. Insensitive and asensitive cursors are not updatable.
- **DESCRIBE INTO DESCRIPTOR** – the prepared statement is described into the specified descriptor. The describe type may be any of the describe types allowed in the **DESCRIBE** statement.

The DESCRIBE INTO DESCRIPTOR clause might improve performance, as it decrease the required client/server communication.

- **WITH EXECUTE** – the statement is executed if and only if it is not a **CALL** or **SELECT** statement, and it has no host variables. The statement is immediately dropped after a successful execution. If **PREPARE** and **DESCRIBE** (if any) are successful but the statement cannot be executed, a warning `SQLCODE 111, SQLSTATE 01W08` is set, and the statement is not dropped.

The WITH EXECUTE clause might improve performance, as it decrease the required client/server communication.

- **WITH VARIABLE RESULT** – describes procedures that may have more than one result set, with different numbers or types of columns. If the WITH VARIABLE RESULT clause is used, the database server sets the `SQLCOUNT` value after the describe to one of these values:
 - **0** – the result set may change: the procedure call should be described again following each **OPEN** statement.
 - **1** – the result set is fixed. No redescrbing is required.

Examples

- **Example 1** – prepare a simple query:

```
EXEC SQL PREPARE employee_statement FROM  
'SELECT Surname FROM Employees';
```

Usage

The **PREPARE** statement prepares a SQL statement from the *statement* and associates the prepared statement with *statement-name*.

If a host variable is used for *statement-name*, it must have the type `short int`. There is a typedef for this type in `sqlca.h` called `a_sql_statement_number`. This type is recognized by the SQL preprocessor and can be used in a **DECLARE** section. The host variable is filled in by the database during the **PREPARE** statement and need not be initialized by the programmer.

These statements can be prepared:

- **ALTER**
- **CALL**
- **COMMENT ON**
- **CREATE**
- **DELETE**
- **DROP**
- **GRANT**
- **INSERT**
- **REVOKE**
- **SELECT**
- **SET OPTION**

Preparing **COMMIT**, **PREPARE TO COMMIT**, and **ROLLBACK** statements is still supported for compatibility. However, perform all transaction management operations with static Embedded SQL, because certain application environments may require it. Also, other Embedded SQL systems do not support dynamic transaction management operations.

Note: Make sure that you **DROP** the statement after use. If you do not, then the memory associated with the statement is not reclaimed.

Side Effects

- Any statement previously prepared with the same name is lost.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *DESCRIBE Statement [ESQL]* on page 251
- *DROP Statement* on page 255
- *EXECUTE Statement [ESQL]* on page 276

- *OPEN Statement [ESQL] [SP]* on page 369

PRINT Statement [T-SQL]

Displays a message on the message window of the database server.

Syntax

```
PRINT format-string [, arg-list]
```

Examples

- **Example 1** – displays a message on the server message window:

```
CREATE PROCEDURE print_test
AS
PRINT 'Procedure called successfully'
```

This statement returns the string “Procedure called successfully” to the client:

```
EXECUTE print_test
```

- **Example 2** – uses placeholders in the **PRINT** statement; execute these statements inside a procedure:

```
DECLARE @var1 INT, @var2 INT
SELECT @var1 = 3, @var2 = 5
PRINT 'Variable 1 = %1!, Variable 2 = %2!', @var1, @var2
```

- **Example 3** – uses **RAISERROR** to disallow connections:

```
CREATE procedure DBA.login_check()
begin
  // Allow a maximum of 3 concurrent connections
  IF( db_property('ConnCount') > 3 ) then
    raiserror 28000
    'User %1! is not allowed to connect -- there are
    already %2! users logged on',
    current user,
    cast(db_property('ConnCount') as int)-1;
  ELSE
    call sp_login_environment;
  end if;
end
go
grant execute on DBA.login_check to PUBLIC
go
set option PUBLIC.Login_procedure='DBA.login_check'
go
```

For an alternate way to disallow connections, use the **LOGIN_PROCEDURE** option or the **sp_iqmodifylogin** system stored procedure.

Usage

The **PRINT** statement returns a message to the client window if you are connected from an Open Client application or JDBC application. If you are connected from an Embedded SQL or ODBC application, the message displays on the database server window.

The format string can contain placeholders for the arguments in the optional argument list. These placeholders are of the form *%nn!*, where *nn* is an integer between 1 and 20.

Standards

- SQL—Transact-SQL extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

Must be connected to the database.

See also

- *MESSAGE Statement* on page 366
- *LOGIN_PROCEDURE Option* on page 572

PUT Statement [ESQL]

Inserts a row into the specified cursor.

Syntax

```
PUT cursor-name [ USING DESCRIPTOR sqlda-name
  | FROM hostvar-list ] [ INTO { DESCRIPTOR into-sqlda-name
  | into-hostvar-list } ] [ ARRAY :nnn ]
```

Parameters

- **cursor-name** – identifier or hostvar
- **sqlda-name** – identifier
- **hostvar-list** – may contain indicator variables
- **ARRAY** – can be used to carry out wide puts, which insert more than one row at a time and which might improve performance. The value *nnn* is the number of rows to be inserted. The SQLDA must contain *nnn* * (columns per row) variables. The first row is placed in SQLDA variables 0 to (columns per row) - 1, and so on.

Note: For scroll (values-sensitive) cursors, the inserted row appears if the new row matches the WHERE clause and the keyset cursor has not finished populating. For

dynamic cursors, if the inserted row matches the **WHERE** clause, the row might appear. Inensitive cursors cannot be updated.

Examples

- **Example 1** – use **PUT** in Embedded SQL:

```
EXEC SQL PUT cur_employee FROM :EmployeeID, :Surname;
```

Usage

Inserts a row into the named cursor. Values for the columns are taken from the first **SQLDA** or the host variable list, in a one-to-one correspondence with the columns in the **INSERT** statement (for an **INSERT** cursor) or the columns in the select list (for a **SELECT** cursor).

The **PUT** statement can be used only on a cursor over an **INSERT** or **SELECT** statement that references a single table in the **FROM** clause, or that references an updatable view consisting of a single base table.

If the **sqldata** pointer in the **SQLDA** is the null pointer, no value is specified for that column. If the column has a **DEFAULT VALUE** associated with it, that is used; otherwise, a **NULL** value is used.

The second **SQLDA** or host variable list contains the results of the **PUT** statement.

For information on putting **LONG VARCHAR** or **LONG BINARY** values into the database, see *SET statement [ESQL]*.

Side Effects

- When inserting rows into a value-sensitive (keyset-driven) cursor, the inserted rows appear at the end of the result set, even when they do not match the **WHERE** clause of the query or if an **ORDER BY** clause would normally have placed them at another location in the result set.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

Requires **INSERT** privilege.

See also

- *DELETE (positioned) Statement [ESQL] [SP]* on page 250
- *INSERT Statement* on page 327
- *SET Statement [ESQL]* on page 426

- *UPDATE Statement* on page 448
- *UPDATE (positioned) Statement [ESQL] [SP]* on page 451

RAISERROR Statement [T-SQL]

Allows user-defined errors to be signaled, and sends a message on the client.

Syntax

```
RAISERROR error-number [ format-string ] [, arg-list ]
```

Parameters

- **error-number** – a 5-digit integer greater than 17000. The error number is stored in the global variable @@error.
- **format-string** – if not supplied or is empty, the error number is used to locate an error message in the system tables. Adaptive Server Enterprise obtains messages 17000-19999 from the SYSMESSAGES table. In SAP Sybase IQ, this table is an empty view, so errors in this range should provide a format string. Messages for error numbers of 20000 or greater are obtained from the SYS.SYSUSERMESSAGES table.

The *format-string* can be up to 255 bytes long. This is the same as in Adaptive Server Enterprise.

The format string can contain placeholders for the arguments in the optional argument list. These placeholders are of the form %nn!, where nn is an integer between 1 and 20.

Examples

- **Example 1** – raises error 99999, which is in the range for user-defined errors, and send a message to the client:

```
RAISERROR 99999 'Invalid entry for this
column: %1!', @val
```

Usage

There is no comma between the *error-number* and the *format-string* parameters. The first item following a comma is interpreted as the first item in the argument list.

The extended values supported by the SQL Server or Adaptive Server Enterprise **RAISERROR** statement are not supported in SAP Sybase IQ.

Intermediate RAISERROR status and code information is lost after the procedure terminates. If at return time an error occurs along with the RAISERROR, then the error information is returned and the RAISERROR information is lost. The application can query intermediate RAISERROR statuses by examining @@error global variable at different execution points.

Standards

- SQL—Transact-SQL extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

Must be connected to the database.

See also

- *CONTINUE_AFTER_RAISERROR Option [TSQL]* on page 498
- *ON_TSQL_ERROR Option [TSQL]* on page 598

READ Statement [Interactive SQL]

Reads Interactive SQL (**dbisql**) statements from a file.

Syntax

```
READ [ ENCODING encoding ] filename [ parameter ] ...
```

Parameters

- **ENCODING** – an identifier or string, which lets you specify the encoding that is used to read the file.

The **READ** statement does not process escape characters when it reads a file. It assumes that the entire file is in the specified encoding. When running Interactive SQL, the encoding that is used to read the data is determined in the following order:

1. The encoding specified by the ENCODING clause (if this clause is specified).
 2. The encoding specified by the byte order mark (BOM) in the file (if a BOM is specified).
 3. The encoding specified with the default_isql_encoding option (if this option is set).
 4. The default encoding for the platform you are running on. On English Windows computers, the default encoding is 1252.
- **filename** – has no file extension. Interactive SQL searches for the same file name with the extension .sql. If filename does not contain an absolute path, Interactive SQL searches for the file. The location of filename is determined based on the location of the READ statement, as follows:
 - If the READ statement is executed directly in Interactive SQL, Interactive SQL first attempts to resolve the path to filename relative to the directory in which Interactive SQL is running. If unsuccessful, Interactive SQL looks for filename in the directories

specified in the environment variable `SQLPATH`, and then the directories specified in the environment variable `PATH`.

- If the `READ` statements reside in an external file (for example, a `.sql` file), Interactive SQL first attempts to resolve the path to filename relative to the location of the external file. If unsuccessful, Interactive SQL looks for filename in a path relative to the directory in which Interactive SQL is running. If still unsuccessful, Interactive SQL looks in the directories specified in the environment variable `SQLPATH`, and then the directories specified in the environment variable `PATH`.
- **parameters** – can be listed after the name of the SQL script file. These parameters correspond to the parameters named in the `PARAMETERS` statement at the beginning of the statement file.

Parameter names must be enclosed in square brackets. Interactive SQL substitutes the corresponding parameter wherever the source file contains { *parameter-name* }, where *parameter-name* is the name of the appropriate parameter.

The parameters passed to a script file can be identifiers, numbers, quoted identifiers, or strings. When quotes are used around a parameter, the quotes are put into the text during the substitution. Parameters that are not identifiers, numbers, or strings (contain spaces or tabs) must be enclosed in square brackets ([]). This allows for arbitrary textual substitution in the script file.

If not enough parameters are passed to the script file, Interactive SQL prompts for values for the missing parameters.

When executing a `reload.sql` file with Interactive SQL, you must specify the encryption key as a parameter. If you do not provide the key in the `READ` statement, Interactive SQL prompts for the key.

Examples

- **Example 1** – this example reads from the fictitious file `status.rpt` and `birthday.sql` and passes the parameter values to the variables within the file:

```
READ status.rpt '160'
READ birthday.sql [ >= '1988-1-1' ] [ <= '1988-1-30' ]
```

- **Example 2** –

```
[test1.sql]
PARAMETERS par1, par2;

BEGIN
DECLARE v_par1 int;
DECLARE v_par2 varchar(200)

SET v_par1 = {par1};
SET v_par2 = {par2};

MESSAGE STRING('PAR1 Value: ', v_par1 ) TO CLIENT;
```

SQL Statements

```
MESSAGE STRING('PAR2 Value: ', v_par2 ) TO CLIENT;

END;

(USR1)> READ test1.sql 123 '041028'
PAR1 Value: 123
PAR2 Value: 041028
```

Note: The second parameter value 041028 must be enclosed in quotes, as *v_par2* is declared as a character data type.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *DEFAULT_ISQL_ENCODING Option [Interactive SQL]* on page 520
- *PARAMETERS Statement [Interactive SQL]* on page 374

REFRESH TEXT INDEX Statement

Refreshes a text index.

Syntax

```
REFRESH TEXT INDEX text-index-name ON [ owner. ] table-name
  [ WITH {
    ISOLATION LEVEL isolation-level
    | EXCLUSIVE MODE
    | SHARE MODE } ]
  [ FORCE { BUILD | INCREMENTAL } ]
```

Parameters

- **WITH** – use the WITH clause to specify what kind of locks to obtain on the underlying base tables during the refresh. The types of locks obtained determine how the text index is populated and how concurrency for transactions is affected. If you do not specify the WITH clause, the default is WITH ISOLATION LEVEL READ UNCOMMITTED, regardless of any isolation level set for the connection.

You can specify the following WITH clause options:

- **ISOLATION LEVEL isolation-level** – use `WITH ISOLATION LEVEL` to change the isolation level for the execution of the refresh operation. The original isolation level of the connection is restored at the end of the statement execution.
- **EXCLUSIVE MODE** – use `WITH EXCLUSIVE MODE` if you do not want to change the isolation level, but want to guarantee that the data is updated to be consistent with committed data in the underlying table. When using `WITH EXCLUSIVE MODE`, exclusive table locks are placed on the underlying base table and no other transaction can execute queries, updates, or any other action against the underlying table(s) until the refresh operation is complete. If table locks cannot be obtained, the refresh operation fails and an error is returned.
- **SHARE MODE** – use `WITH SHARE MODE` to give read access on the underlying table to other transactions while the refresh operation takes place. When this clause is specified, shared table locks are obtained on the underlying base table before the refresh operation is performed and are held until the refresh operation completes.
- **FORCE** – use this clause to specify the refresh method. If this clause is not specified, the database server decides whether to do an incremental update or a full rebuild based on how much of the table has changed.
 - **FORCE BUILD** – refreshes the text index by recreating it. Use this clause to force a complete rebuild of the text index.
 - **FORCE INCREMENTAL** – refreshes the text index based only on what has changed in the underlying table. An incremental refresh takes less time to complete if there have not been a significant amount of updates to the underlying table. Use this clause to force an incremental update of the text index.

An incremental refresh does not remove deleted entries from the text index. As a result, the size of the text index may be larger than expected to contain the current and historic data. Typically, this issue occurs with text indexes that are always manually refreshed with the `FORCE INCREMENTAL` clause. On automatically refreshed text indexes, historic data is automatically deleted when it makes up 50% of the total size of the text index.

Examples

- **Example 1** – refreshes a fictitious text index called `MarketingTextIndex`, forcing it to be rebuilt:

```
REFRESH TEXT INDEX MarketingTextIndex ON
GROUPO.MarketingInformation
FORCE BUILD;
```

Usage

This statement can only be used on text indexes defined as `MANUAL REFRESH` or `AUTO REFRESH`.

SQL Statements

When using the FORCE clause, you can examine the results of the **sa_text_index_stats** system procedure to decide whether a complete rebuild (FORCE BUILD), or incremental update (FORCE INCREMENTAL) is most appropriate.

You cannot execute the REFRESH TEXT INDEX statement on a text index that is defined as IMMEDIATE REFRESH.

For MANUAL REFRESH text indexes, use the **sa_text_index_stats** system procedure to determine whether the text index should be refreshed. Divide pending_length by doc_length, and use the percentage as a guide for deciding whether a refresh is required. To determine the type of rebuild required, use the same process for deleted_length and doc_count.

This statement cannot be executed when there are cursors opened with the WITH HOLD clause that use either statement or transaction snapshots.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- REFERENCES privilege on the table.
- You own the table.

RELEASE SAVEPOINT Statement

Releases a savepoint within the current transaction.

Syntax

```
RELEASE SAVEPOINT [ savepoint-name ]
```

Parameters

- **savepoint-name** – an identifier specified on a **SAVEPOINT** statement within the current transaction. If *savepoint-name* is omitted, the most recent savepoint is released.

Usage

Releasing a savepoint does not perform any type of **COMMIT**; it simply removes the savepoint from the list of currently active savepoints.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise. A similar feature is available in an Adaptive Server Enterprise-compatible manner using nested transactions.

Permissions

There must have been a corresponding **SAVEPOINT** within the current transaction.

See also

- *ROLLBACK TO SAVEPOINT Statement* on page 413
- *SAVEPOINT Statement* on page 415

REMOVE Statement

Removes a class, a package, or a JAR file from a database. Removed classes are no longer available for use as a variable type. Any class, package, or JAR to be removed must already be installed.

Syntax

```
REMOVE JAVA classes_to_remove
```

```
classes_to_remove:
{ CLASS java_class_name [, java_class_name ]...
| PACKAGE java_package_name [, java_package_name ]...
| JAR jar_name [, jar_name ]... [ RETAIN CLASSES ] }
```

Parameters

- **java_class_name** – the name of one or more Java classes to be removed. Those classes must be installed classes in the current database.
- **java_package_name** – the name of one or more Java packages to be removed. Those packages must be the name of packages in the current database.
- **jar_name** – a character string value of maximum length 255. Each *jar_name* must be equal to the *jar_name* of a retained JAR in the current database. Equality of *jar_name* is determined by the character string comparison rules of the SQL system.
- **RETAIN CLASSES** – the specified JARs are no longer retained in the database, and the retained classes have no associated JAR. If **RETAIN CLASSES** is specified, this is the only action of the **REMOVE** statement.

Examples

- **Example 1** – remove a Java class named “Demo” from the current database:

```
REMOVE JAVA CLASS Demo
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise. A similar feature is available in an Adaptive Server Enterprise-compatible manner using nested transactions.

Permissions

Requires one of:

- MANAGE ANY EXTERNAL OBJECT system privilege.
- You own the object.

RESIGNAL Statement

Resignals an exception condition.

Syntax

```
RESIGNAL [ exception-name ]
```

Examples

- **Example 1** – this code fragment returns all exceptions except for “Column Not Found” to the application:

```
...  
DECLARE COLUMN_NOT_FOUND EXCEPTION  
    FOR SQLSTATE '52003';  
...  
EXCEPTION  
WHEN COLUMN_NOT_FOUND THEN  
    SET message='Column not found' ;  
WHEN OTHERS THEN  
    RESIGNAL ;
```

Usage

Within an exception handler, **RESIGNAL** lets you quit the compound statement with the exception still active, or quit reporting another named exception. The exception is handled by another exception handler or returned to the application. Any actions by the exception handler before the **RESIGNAL** are undone.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported in Adaptive Server Enterprise. Error handling in Transact-SQL procedures is carried out using the **RAISERROR** statement.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84
- *SIGNAL Statement* on page 437

RESTORE DATABASE Statement

Restores an SAP Sybase IQ database backup from one or more archive devices.

Syntax

Syntax 1

```
RESTORE DATABASE 'db_file'
    'archive_device' [ FROM 'archive_device' ]...
... [ CATALOG ONLY ]
... [ KEY key_spec ]
... [ [ RENAME logical-dbfile-name TO 'new-dbspace-path']...
    | VERIFY [ COMPATIBLE ] ]
```

Syntax 2

```
RESTORE DATABASE 'database-name'
    [ restore-option ... ]
    FROM 'archive_device' ...
```

restore-option:

```
  READONLY dbspace-or-file [, ... ]
  KEY key_spec
  RENAME file-name TO new-file-path ...
```

Parameters

- **db_file** – relative or absolute path of the database to be restored. Can be the original location, or a new location for the catalog store file.
- **key_spec** – quoted string including mixed cases, numbers, letters, and special characters. It might be necessary to protect the key from interpretation or alteration by the command shell.
- **FROM** – specifies the name of the *archive_device* from which you are restoring, delimited with single quotation marks. If you are using multiple archive devices, specify them using

separate FROM clauses. A comma-separated list is not allowed. Archive devices must be distinct. The number of FROM clauses determines the amount of parallelism SAP Sybase IQ attempts with regard to input devices.

The backup/restore API DLL implementation lets you specify arguments to pass to the DLL when opening an archive device. For third-party implementations, the *archive_device* string has this format:

```
'DLLIdentifier::vendor_specific_information'
```

A specific example is:

```
'spsc::workorder=12;volname=ASD002'
```

The *archive_device* string length can be up to 1023 bytes. The *DLLIdentifier* portion must be 1 to 30 bytes in length and can contain only alphanumeric and underscore characters. The *vendor_specific_information* portion of the string is passed to the third-party implementation without checking its contents.

Note: Only certain third-party products are certified with SAP Sybase IQ using this syntax. See the *Release Bulletin* for additional usage instructions or restrictions. Before using any third-party product to back up your SAP Sybase IQ database, make sure it is certified. See the *Release Bulletin*, or see the Sybase Certification Reports for the SAP Sybase IQ product in *Technical Documents*.

For the Sybase implementation of the backup/restore API, you need not specify information other than the tape device name or file name. However, if you use disk devices, you must specify the same number of archive devices on the **RESTORE** as given on the backup; otherwise, you may have a different number of restoration devices than the number used to perform the backup. A specific example of an archive device for the Sybase API DLL that specifies a nonrewinding tape device for a UNIX system is:

```
 '/dev/rmt/0n'
```

- **CATALOG ONLY** – restores only the backup header record from the archive media.
- **RENAME** – restore one or more SAP Sybase IQ database files to a new location. Specify each *dbspace-name* you are moving as it appears in the *SYSFILE* table. Specify *new-dbpace-path* as the new raw partition, or the new full or relative path name, for that *dbspace*.

If relative paths were used to create the database files, the files are restored by default relative to the catalog store file (the *SYSTEM* *dbspace*), and a rename clause is not required. If absolute paths were used to create the database files and a rename clause is not specified for a file, it is restored to its original location.

Relative path names in the **RENAME** clause work as they do when you create a database or *dbspace*: the main IQ store *dbspace*, temporary store *dbspaces*, and Message Log are restored relative to the location of *db_file* (the catalog store); user-created IQ store *dbspaces* are restored relative to the directory that holds the main IQ *dbspace*.

Do not use the RENAME clause to move the SYSTEM dbspace, which holds the catalog store. To move the catalog store, and any files created relative to it and not specified in a RENAME clause, specify a new location in the *db_file* parameter.

- **VERIFY [COMPATIBLE** – directs the server to validate the specified SAP Sybase IQ database backup archives for a full, incremental, incremental since full, or virtual backup. The backup must be SAP Sybase IQ version 12.6 or later. The verification process checks the specified archives for the same errors a restore process checks, but performs no write operations. All status messages and detected errors are written to the server log file.

You cannot use the RENAME clause with the VERIFY clause; an error is reported.

The backup verification process can run on a different host than the database host. You must have the BACKUP DATABASE system privilege to run RESTORE VERIFY.

If the COMPATIBLE clause is specified with VERIFY, the compatibility of an incremental archive is checked with the existing database files. If the database files do not exist on the system on which **RESTORE...VERIFY COMPATIBLE** is invoked, an error is returned. If COMPATIBLE is specified while verifying a full backup, the keyword is ignored; no compatibility checks need to be made while restoring a full backup.

You must have the database and log files (.db and .log) to validate the backup of a read-only dbspace within a full backup. If you do not have these files, validate the entire backup by running **RESTORE...VERIFY** without the READONLY *dbspace* clause.

Note: The verification of a backup archive is different than the database consistency checker (DBCC) verify mode (`sp_iqcheckdb 'verify...'`). RESTORE VERIFY validates the consistency of the backup archive to be sure it can be restored, whereas DBCC validates the consistency of the database data.

Run `sp_iqcheckdb 'verify...'` before taking a backup. If an inconsistent database is backed up, then restored from the same backup archive, the data continues to be in an inconsistent state, even if **RESTORE VERIFY** reports a successful validation.

Examples

- **Example 1** – this UNIX example restores the `iqdemo` database from tape devices `/dev/rmt/0` and `/dev/rmt/2` on a Sun Solaris platform. On Solaris, a RESTORE from tape must specify the use of the rewinding device. Therefore, do not include the letter 'n' after the device name, which specifies “no rewind on close.” To specify this feature with **RESTORE**, use the naming convention appropriate for your UNIX platform. (Windows does not support this feature.)

```
RESTORE DATABASE 'iqdemo'
FROM '/dev/rmt/0'
FROM '/dev/rmt/2'
```

- **Example 2** – restore an encrypted database named `marvin` that was encrypted with the key `is!seCret`:

```
RESTORE DATABASE 'marvin'
FROM 'marvin_bkup_file1'
FROM 'marvin_bkup_file2'
FROM 'marvin_bkup_file3'
KEY 'is!seCrét'
```

- **Example 3** – this example shows the syntax of a **BACKUP** statement and two possible **RESTORE** statements. (This example uses objects in the `iqdemo` database for illustration purposes. Note that `iqdemo` includes a sample user dbspace named `iq_main` that may not be present in your database.)

Given this **BACKUP** statement:

```
BACKUP DATABASE READONLY DBSPACES iq_main
TO '/system1/IQ16/demo/backup/iqmain'
```

The dbspace `iq_main` can be restored using either of these **RESTORE** statements:

```
RESTORE DATABASE 'iqdemo' READONLY DBSPACES iq_main
FROM '/system1/IQ16/demo/backup/iqmain'
```

or

```
RESTORE DATABASE 'iqdemo'
FROM '/system1/IQ16/demo/backup/iqmain'
```

A selective backup backs up either all **READWRITE** dbspaces or specific read-only dbspaces or dbfiles. Selective backups are a subtype of either full or incremental backups.

Notes:

- You can take a **READONLY** selective backup and restore all objects from this backup (as in the second example above).
- You can take an all-inclusive backup and restore read-only files and dbspaces selectively.
- You can take a **READONLY** selective backup of multiple read-only files and dbspaces and restore a subset of read-only files and dbspaces selectively. See *Permissions*.
- You can restore the read-only backup, only if the read-only files have not changed since the backup. Once the dbspace is made read-write again, the read-only backup is invalid, unless you restore the entire read-write portion of the database back to the point at which the read-only dbspace was read-only.
- Decide which backup subtype to use (either selective or non-selective) and use it consistently. If you must switch from a non-selective to a selective backup, or vice versa, always take a non-selective full backup before switching to the new subtype, to ensure that you have all changes.
- **Example 4** – syntax to validate the database archives using the **VERIFY** clause, without performing any write operations:

```
RESTORE DATABASE <database_name.db>
FROM '/sys1/dump/dmp1'
FROM '/sys1/dump/dmp2'
VERIFY
```

When you use `validate`, specify a different database name to avoid Database name not unique errors. If the original database is `iqdemo.db`, for example, use `iq_demo_new.db` instead:

```
RESTORE DATABASE iqdemo_new.db FROM iqdemo.bkp VERIFY
```

Usage

The **RESTORE** command requires exclusive access by a user with the **SERVER OPERATOR** system privilege to the database. This exclusive access is achieved by setting the **-gd** switch to **DBA**, which is the default when you start the server engine.

Issue the **RESTORE** command before you start the database (you must be connected to the `utility_db` database). Once you finish specifying **RESTORE** commands for the type of backup, that database is ready to be used. The database is left in the state that existed at the end of the first implicit **CHECKPOINT** of the last backup you restored. You can now specify a **START DATABASE** to allow other users to access the restored database.

The maximum size for a complete **RESTORE** command, including all clauses, is 32KB.

When restoring to a raw device, make sure the device is large enough to hold the dbspace you are restoring. SAP Sybase IQ **RESTORE** checks the raw device size and returns an error, if the raw device is not large enough to restore the dbspace.

BACKUP allows you to specify full or incremental backups. There are two kinds of incremental backups. **INCREMENTAL** backs up only those blocks that have changed and committed since the last backup of any type (incremental or full). **INCREMENTAL SINCE FULL** backs up all the blocks that have changed since the last full backup. If a **RESTORE** of a full backup is followed by one or more incremental backups (of either type), no modifications to the database are allowed between successive **RESTORE** commands. This rule prevents a **RESTORE** from incremental backups on a database in need of crash recovery, or one that has been modified. You can still overwrite such a database with a **RESTORE** from a full backup.

Before starting a full restore, you must delete two files: the catalog store file (default name `dbname.db`) and the transaction log file (default name `dbname.log`).

If you restore an incremental backup, **RESTORE** ensures that backup media sets are accessed in the proper order. This order restores the last full backup tape set first, then the first incremental backup tape set, then the next most recent set, and so forth, until the most recent incremental backup tape set. If a user with the **SERVER OPERATOR** system privilege produced an **INCREMENTAL SINCE FULL** backup, only the full backup tape set and the most recent **INCREMENTAL SINCE FULL** backup tape set is required; however, if there is an **INCREMENTAL** backup made since the **INCREMENTAL SINCE FULL** backup, it also must be applied.

SAP Sybase IQ ensures that the restoration order is appropriate, or it displays an error. Any other errors that occur during the restore results in the database being marked corrupt and unusable. To clean up a corrupt database, do a **RESTORE** from a full backup, followed by any

SQL Statements

additional incremental backups. Since the corruption probably happened with one of those backups, you might need to ignore a later backup set and use an earlier set.

To restore read-only files or dbspaces from an archive backup, the database may be running and the administrator may connect to the database when issuing the **RESTORE** statement. The read-only file pathname need not match the names in the backup, if they otherwise match the database system table information.

The database must not be running to restore a **FULL, INCREMENTAL SINCE FULL**, or **INCREMENTAL** restore of either a **READWRITE FILES ONLY** or an all files backup. The database may or may not be running to restore a backup of read-only files. When restoring specific files in a read-only dbspace, the dbspace must be offline. When restoring read-only files in a read-write dbspace, the dbspace can be online or offline. The restore closes the read-only files, restores the files, and reopens those files at the end of the restore.

You can use selective restore to restore a read-only dbspace, as long as the dbspace is still in the same read-only state.

Other **RESTORE** issues:

- **RESTORE** to disk does not support raw devices as archival devices.
- SAP Sybase IQ does not rewind tapes before using them; on rewinding tape devices, it does rewind tapes after using them. You must position each tape to the start of the SAP Sybase IQ data before starting the **RESTORE**.
- During **BACKUP** and **RESTORE** operations, if SAP Sybase IQ cannot open the archive device (for example, when it needs the media loaded) and the **ATTENDED** option is **ON**, it waits for ten seconds for you to put the next tape in the drive, and then tries again. It continues these attempts indefinitely until either it is successful or the operation is terminated with Ctrl+C.
- If you press Ctrl+C, **RESTORE** fails and returns the database to its state before the restoration began.
- If disk striping is used, the striped disks are treated as a single device.
- The `file_name` column in the `SYSFILE` system table for the `SYSTEM` dbspace is not updated during a restore. For the `SYSTEM` dbspace, the `file_name` column always reflects the name when the database was created. The file name of the `SYSTEM` dbspace is the name of the database file.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

The permissions required to execute this statement are set using the **-gu** server command line option, as follows:

- **NONE** – No user can issue this statement.
- **DBA** – Requires the SERVER OPERATOR system privilege.
- **UTILITY_DB** – Only those users who can connect to the `utility_db` database can issue this statement.

See also

- *BACKUP Statement* on page 78

RESUME Statement

Resumes execution of a procedure that returns result sets.

Syntax

Syntax 1 – Supported in **dbisqlc**.

```
RESUME cursor-name
```

Syntax 2 – Supported in **dbisql**.

```
RESUME [ ALL ]
```

Parameters

- **cursor-name** – identifier or host-variable

Examples

- **Example 1** – embedded SQL examples:

```
EXEC SQL RESUME cur_employee;
```

and

```
EXEC SQL RESUME :cursor_var;
```

- **Example 2** – **dbisql** example:

```
CALL sample_proc() ;
RESUME ALL;
```

Usage

The procedure executes until the next result set (**SELECT** statement with no INTO clause) is encountered. If the procedure completes and no result set is found, the `SQLSTATE_PROCEDURE_COMPLETE` warning is set. This warning is also set when you **RESUME** a cursor for a **SELECT** statement.

SQL Statements

Syntax 1 – supported in **dbisqlc** but not **dbisql** (Interactive SQL) or when connected to the database using the SQL Anywhere JDBC driver.

Syntax 2 – supported in **dbisql**. Resumes the current procedure. If ALL is not specified, executing **RESUME** displays the next result set or, if no more result sets are returned, completes the procedure. In **dbisql**, the **RESUME ALL** statement cycles through all result sets in a procedure, without displaying them, and completes the procedure. This is useful mainly in testing procedures.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise.

Permissions

The cursor must have been previously opened.

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240

RETURN Statement

Exits a function or procedure unconditionally, optionally providing a return value. Statements following **RETURN** are not executed.

Syntax

```
RETURN [ ( expression ) ]
```

Parameters

- **expression** – if supplied, the value of *expression* is returned as the value of the function or procedure.

Within a function, the expression should be of the same data type as the RETURN data type of the function.

Examples

- **Example 1** – returns the product of three numbers:

```
CREATE FUNCTION product ( a numeric,  
                          b numeric ,  
                          c numeric)  
RETURNS numeric  
BEGIN
```

```
RETURN ( a * b * c ) ;
END
```

- **Example 2** – calculates the product of three numbers:

```
SELECT product (2, 3, 4)
```

```
product (2,3,4)
24
```

- **Example 3** – avoids executing a complex query, if it is meaningless:

```
CREATE PROCEDURE customer_products
( in customer_id integer DEFAULT NULL)
RESULT ( id integer, quantity_ordered integer )
BEGIN
  IF customer_id NOT IN (SELECT ID FROM Customers)
  OR customer_id IS NULL THEN
    RETURN
  ELSE
    SELECT ID, sum(
      SalesOrderItems.Quantity )
    FROM Products,
      SalesOrderItems,
      SalesOrders
    WHERE SalesOrders.CustomerID = customer_id
    AND SalesOrders.ID = SalesOrderItems.ID
    AND SalesOrderItems.ProductID = Products.ID
    GROUP BY Products.ID
  END IF
END
```

Usage

RETURN is used in procedures for Transact-SQL-compatibility, and is used to return an integer error code.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Transact-SQL procedures use the return statement to return an integer error code.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84
- *CREATE PROCEDURE Statement* on page 165

REVOKE CHANGE PASSWORD Statement

Removes the ability of a user to manage passwords and administer the system privilege.

Syntax

```
REVOKE [ ADMIN OPTION FOR ] CHANGE PASSWORD
  [(target_user_list
   | ANY
   | ANY WITH ROLES target_role_list )]
FROM userID [,...]
```

Parameters

- **target_user_list** – users the grantee has the potential to impersonate. The list must consist of existing users or user-extended roles with login passwords. Separate the userIDs in the list with commas.
- **ANY** – all database users with login passwords become potential target users to manage passwords for each grantee.
- **ANY WITH ROLES target_role_list** – list of target roles for each grantee. Any users who are granted any of the target roles become potential target users for each grantee. The *target_role_list* must consist of existing roles and the users who are granted said roles must consist of database users with login passwords. Use commas to separate multiple userIDs.
- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples

- **Example 1** – removes the ability of Joe to manage the passwords of Sally or Bob:

```
REVOKE CHANGE PASSWORD (Sally, Bob) FROM Joe
```
- **Example 2** – if the CHANGE PASSWORD system privilege was originally granted to Sam with the WITH ADMIN OPTION clause, this example removes the ability of Sam to grant the CHANGE PASSWORD system privilege to another user, but still allows Sam to manage passwords for those users specified in the original **GRANT CHANGE PASSWORD** statement. However, if the CHANGE PASSWORD system privilege was originally granted to Sam with the WITH ADMIN ONLY OPTION clause, this example removes all permissions to the system privilege from Sam.

```
REVOKE ADMIN OPTION FOR CHANGE PASSWORD FROM Sam
```

Usage

Depending on how the CHANGE PASSWORD system privilege was initially granted, using the ADMIN OPTION FOR clause when revoking the CHANGE PASSWORD system

privilege has different results. If the **CHANGE PASSWORD** system privilege was originally granted with the **WITH ADMIN OPTION** clause, including the **ADMIN OPTION FOR** clause in the revoke statement revokes only the ability to administer the **CHANGE PASSWORD** system privilege (that is, grant the system privilege to another user). The ability to actually manage passwords for other users remains. However, if the **CHANGE PASSWORD** system privilege was originally granted with the **WITH ADMIN ONLY OPTION** clause, including the **ADMIN OPTION FOR** clause in the revoke statement is semantically equivalent to revoking the entire **CHANGE PASSWORD** system privilege. Finally, if the **CHANGE PASSWORD** system privilege was originally granted with the **WITH NO ADMIN OPTION** clause, and the **ADMIN OPTION FOR** clause is included in the revoke statement, nothing is revoked because there were no administrative rights granted in the first place.

You can revoke the **CHANGE PASSWORD** system privilege from any combination of users and roles granted.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the **CHANGE PASSWORD** system privilege granted with administrative rights.

See also

- *GRANT CHANGE PASSWORD Statement* on page 304

REVOKE CONNECT Statement

Removes a user from the database.

Syntax

```
REVOKE CONNECT  
FROM userID [, ...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Usage

Use system procedures or **CREATE USER** and **DROP USER** statements, not **GRANT** and **REVOKE** statements, to add and remove user IDs.

You cannot revoke the connect privileges from a user if he or she owns database objects, such as tables. Attempting to do so with a **REVOKE** statement, or **sp_droprologin** or **sp_iqdroprologin**

SQL Statements

stored procedure returns an error such as Cannot drop a user that owns tables in runtime system.

Standards

ANSI SQL – compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE ANY USER system privilege.

Note: If revoking **CONNECT** permissions or revoking table permissions from another user, the target user cannot be connected to the database.

See also

- *GRANT CONNECT Statement* on page 306

REVOKE CREATE Statement

Removes CREATE permissions on the specified dbspace from the specified user IDs.

Syntax

```
REVOKE CREATE ON dbspace-name  
FROM userID [,...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Examples

- **Example 1** – revokes the CREATE privilege on dbspace DspHist from user Smith:

```
REVOKE CREATE ON DspHist FROM Smith
```

- **Example 2** – revokes the CREATE permission on dbspace DspHist from user ID fionat from the database:

```
REVOKE CREATE ON DspHist FROM fionat
```

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the MANAGE ANY DBSPACE system privilege.

See also

- *GRANT CREATE Statement* on page 308

REVOKE EXECUTE Statement

Removes EXECUTE permissions that were given using the **GRANT** statement.

Syntax

```
REVOKE EXECUTE ON [ owner. ] procedure-name
FROM userID [ , ... ]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL—Syntax is a Persistent Stored Module feature.
- Sybase—Syntax is supported by Adaptive Server Enterprise. User management and security models are different for v and SAP Sybase IQ.

Permissions

You either:

- Own the procedure, or
- Have been granted the MANAGE ANY OBJECT PRIVILEGE system privilege.

See also

- *GRANT EXECUTE Statement* on page 310

REVOKE INTEGRATED LOGIN Statement

Removes the INTEGRATED LOGIN permissions that were given using the **GRANT** statement.

Syntax

```
REVOKE INTEGRATED LOGIN
FROM userID [ , ... ]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *GRANT INTEGRATED LOGIN Statement* on page 311

REVOKE KERBEROS LOGIN Statement

Removes KERBEROS LOGIN permissions that were given using the **GRANT** statement.

Syntax

```
REVOKE KERBEROS LOGIN  
FROM userID [, ...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the `MANAGE ANY USER` system privilege.

See also

- *GRANT KERBEROS LOGIN Statement* on page 312

REVOKE Object-Level Privilege Statement

Removes object-level privileges that were given using the **GRANT** statement.

Syntax

```
REVOKE { object-level-privilege [, ...]
        [ owner.]table-name
        FROM userID [, ...]
```

object-level-privilege:

```
ALL [ PRIVILEGES ]
| ALTER
| DELETE
| INSERT
| LOAD
| REFERENCE [ ( column-name [, ...] ) ]
| SELECT [ ( column-name [, ...] ) ]
| TRUNCATE
| UPDATE [ ( column-name, ... ) ] }
```

Parameters

- **userID** – must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate the userIDs in the list with commas.
- **ALL** – grants all privileges to users
- **ALTER** – users can alter this table with the **ALTER TABLE** statement. This privilege is not allowed for views.
- **DELETE** – users can delete rows from this table or view.
- **INSERT** – users can insert rows into the named table or view.
- **LOAD** – users can load data into the named table or view.
- **REFERENCES** – users can create indexes on the named tables, and foreign keys that reference the named tables. If column names are specified, then users can reference only those columns. REFERENCES privileges on columns cannot be granted for views, only for tables.
- **SELECT** – users can look at information in this view or table. If column names are specified, then the users can look at only those columns. SELECT permissions on columns cannot be granted for views, only for tables.
- **TRUNCATE** – users can truncate the named table or view.
- **UPDATE** – users can update rows in this view or table. If column names are specified, users can update only those columns. UPDATE privileges on columns cannot be granted

SQL Statements

for views, only for tables. To update a table, users must have both SELECT and UPDATE privilege on the table.

Examples

- **Example 1** – prevents user Dave from inserting into the Employees table:

```
REVOKE INSERT ON Employees FROM Dave
```

- **Example 2** – prevents user Dave from updating the Employees table:

```
REVOKE UPDATE ON Employees FROM Dave
```

Standards

- SQL–Syntax is an entry-level feature.
- Sybase–Syntax is supported in Adaptive Server Enterprise.

Permissions

You either:

- Own the table, or
- Have the MANAGE ANY OBJECT PRIVILEGE system privilege granted with the GRANT OPTION clause.

See also

- *GRANT Object-Level Privilege Statement* on page 309

REVOKE ROLE Statement

Removes a users membership in a role or his or her ability to administer the role.

Syntax

```
REVOKE [ ADMIN OPTION FOR ] ROLE role_name [, ...]  
FROM grantee [, ...]
```

role_name:

```
dbo+++  
| diagnostics+++  
| PUBLIC+++  
| rs_systabgroup+++  
| SA_DEBUG+++  
| SYS+++  
| SYS_AUTH_SA_ROLE  
| SYS_AUTH_SSO_ROLE  
| SYS_AUTH_DBA_ROLE  
| SYS_AUTH_RESOURCE_ROLE  
| SYS_AUTH_BACKUP_ROLE
```

```

| SYS_AUTH_VALIDATE_ROLE
| SYS_AUTH_WRITEFILE_ROLE
| SYS_AUTH_WRITEFILECLIENT_ROLE
| SYS_AUTH_READFILE_ROLE
| SYS_AUTH_READFILECLIENT_ROLE
| SYS_AUTH_PROFILE_ROLE
| SYS_AUTH_USER_ADMIN_ROLE
| SYS_AUTH_SPACE_ADMIN_ROLE
| SYS_AUTH_MULTIPLEX_ADMIN_ROLE
| SYS_AUTH_OPERATOR_ROLE
| SYS_AUTH_PERMS_ADMIN_ROLE
| SYS_REPLICATE_ADMIN_ROLE†††
| SYS_RUN_REPLICATE_ROLE†††
| SYS_SPATIAL_ADMIN_ROLE†††
| user-defined role name

```

^{†††}The ADMIN OPTION FOR clause is not supported for system roles.

Parameters

- **role_name** – must already exist in the database. Separate multiple role names with commas.
- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
- **ADMIN OPTION FOR** – each *userID* must have been granted administrative privilege over the specified *role_name*.

Note: This clause revokes administrative privileges of the role only, not membership in the role, unless the role was originally granted with the WITH ADMIN ONLY OPTION clause. For roles granted with the WITH ADMIN ONLY OPTION clause, the ADMIN OPTION FOR clause is optional as it is semantically equivalent to revoking membership in a role in its entirety.

Examples

- **Example 1** – revokes the user-defined (standalone) role Role1 from User1:

```
REVOKE ROLE Role1 FROM User1
```

After you execute this command, User1 no longer has the rights to perform any authorized tasks using any system privileges granted to Role1.

- **Example 2** – revokes the ability for User1 to administer the compatibility role SYS_AUTH_WRITEFILE_ROLE:

```
REVOKE ADMIN OPTION FOR ROLE SYS_AUTH_WRITEFILE_ROLE FROM User1
```

User1 retains the ability to perform any authorized tasks granted by SYS_AUTH_WRITEFILE_ROLE.

Standards

- SQL – Other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – Syntax is supported in Adaptive Server Enterprise.

Permissions

Requires the MANAGE ROLES system privilege to revoke these roles:

- diagnostics
- dbo
- PUBLIC
- rs_systabgroup
- SA_DEBUG
- SYS
- SYS_RUN_REPLICATE_ROLE
- SYS_SPATIAL_ADMIN_ROLE

Requires administrative privilege over the role to revoke these roles:

- SYS_AUTH_SA_ROLE
- SYS_AUTH_SSO_ROLE
- SYS_AUTH_DBA_ROLE
- SYS_AUTH_RESOURCE_ROLE
- SYS_AUTH_BACKUP_ROLE
- SYS_AUTH_VALIDATE_ROLE
- SYS_AUTH_WRITEFILE_ROLE
- SYS_AUTH_WRITEFILECLIENT_ROLE
- SYS_AUTH_READFILE_ROLE
- SYS_AUTH_READFILECLIENT_ROLE
- SYS_AUTH_PROFILE_ROLE
- SYS_AUTH_USER_ADMIN_ROLE
- SYS_AUTH_SPACE_ADMIN_ROLE
- SYS_AUTH_MULTIPLEX_ADMIN_ROLE
- SYS_AUTH_OPERATOR_ROLE
- SYS_AUTH_PERMS_ADMIN_ROLE
- <user-defined role name>

See also

- *GRANT ROLE Statement* on page 312
- *GRANT System Privilege Statement* on page 319

REVOKE SET USER Statement

Removes the ability for one user to impersonate another user and to administer the SET USER system privilege.

Syntax

```
REVOKE [ ADMIN OPTION FOR ] SETUSER
    (target_user_list
     | ANY
     | ANY WITH ROLES target_role_list ])
FROM userID [,...]
```

Parameters

- **target_user_list** – must consist of existing users with login passwords and is the potential list of target users who can no longer be impersonated by grantee users. Separate the user IDs in the list with commas.
- **ANY** – the potential list of target users for each grantee consists of all database users with login passwords.
- **ANY WITH ROLES target_role_list** – the *target_role_list* must consist of existing roles, and the potential list of target users for each grantee must consist of database users with login passwords that have a subset of roles in *target_role_list*. Separate the list of roles with commas.
- **userID** – each *userID* must be the name of an existing user or immutable role. The list must consist of existing users with login passwords. Separate the userIDs in the list with commas.

Examples

- **Example 1** – stops Bob from being able to impersonate Sally or Bob:

```
REVOKE SET USER (Sally, Bob) FROM Bob
```

- **Example 2** – if the SET USER system privilege was originally granted to Sam with the WITH ADMIN OPTION clause, this example removes the ability of Sam to grant the SET USER system privilege to another user, but still allows Sam to impersonate those users already granted to him or her. However, if the SET USER system privilege was originally granted to Sam with the WITH ADMIN ONLY OPTION clause, this example removes all permissions to the system privilege from Sam.

```
REVOKE ADMIN OPTION FOR SET USER FROM Sam
```

Usage

Depending on how the SET USER system privilege was initially granted, using the ADMIN OPTION FOR clause when revoking the SET USER system privilege has different results. If you the SET USER system privilege was originally granted with the WITH ADMIN OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement revokes only the ability to administer the SET USER system privilege (that is, grant the system privilege to another user). The ability to actually impersonate another user remains. However, if the SET USER system privilege was originally granted with the WITH ADMIN ONLY OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement is semantically equivalent to revoking the entire SET USER system privilege. Finally, if the SET USER system privilege was originally grant with the WITH NO ADMIN OPTION clause, and the ADMIN OPTION FOR clause is included in the revoke statement, nothing is revoked because there were no administrative system privileges granted in the first place.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the SET USER system privilege granted with administrative rights.

See also

- *GRANT SET USER Statement* on page 317

REVOKE System Privilege Statement

Removes specific system privileges from specific users and the right to administer the privilege.

Syntax

```
REVOKE [ ADMIN OPTION FOR ] system_privilege [, ...]  
FROM userID [, ...]
```

Parameters

- **system_privilege** – must be an existing system privilege.
- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.
- **ADMIN OPTION FOR** – each *system_privilege* must currently be granted to each *userID* specified with administrative privileges.

Note: This clause revokes only the administrative privileges of the system privilege; the system privilege itself remains granted. However, if the system privilege was originally

granted with the WITH ADMIN ONLY OPTION clause, the ADMIN OPTION FOR clause completely revokes the system privilege. Under this scenario, use of the ADMIN OPTION FOR clause is not required to revoke administrative privileges.

Examples

- **Example 1** – revokes the BACKUP DATABASE system privilege from user Jim:

```
REVOKE BACKUP DATABASE FROM Jim
```

- **Example 2** – assuming the BACKUP DATABASE system privilege was originally granted to user Jim with the WITH ADMIN OPTION clause, this example revokes the ability to administer the BACKUP DATABASE system privilege from user Jim. The ability to perform tasks authorized by the system privilege remains. However, if the BACKUP DATABASE system privilege was originally granted to user Jim with the WITH ADMIN ONLY OPTION clause, this example removes all permissions to the system privilege from user Jim.

```
REVOKE ADMIN OPTION FOR BACKUP DATABASE FROM Jim
```

Usage

Depending on how the system privilege was initially granted, using the ADMIN OPTION FOR clause when revoking a system privilege has different results. If you the system privilege was originally granted with the WITH ADMIN OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement revokes only the ability to administer the system privilege (that is, grant the system privilege to another user). The ability to actually use the system privilege remains. However, if the system privilege was originally granted with the WITH ADMIN ONLY OPTION clause, including the ADMIN OPTION FOR clause in the revoke statement is semantically equivalent to revoking the entire system privilege. Finally, if the system privilege was originally grant with the WITH NO ADMIN OPTION clause, and the ADMIN OPTION FOR clause is included in the revoke statement, nothing is revoked because there were no administrative system privileges granted in the first place.

Standards

- SQL – other syntaxes are vendor extensions to ISO/ANSI SQL grammar.
- Sybase – syntax is not supported by Adaptive Server Enterprise.

Permissions

Requires administrative privilege over the system privilege being revoked.

See also

- *GRANT ROLE Statement* on page 312
- *GRANT System Privilege Statement* on page 319

List of All System Privileges

A list of all system privileges.

System privileges control the rights of users to perform authorized database tasks.

The following is a list of available system privileges:

- ACCESS SERVER LS system privilege
- ALTER ANY INDEX system privilege
- ALTER ANY MATERIALIZED VIEW system privilege
- ALTER ANY OBJECT system privilege
- ALTER ANY OBJECT OWNER system privilege
- ALTER ANY PROCEDURE system privilege
- ALTER ANY SEQUENCE system privilege
- ALTER ANY TABLE system privilege
- ALTER ANY TEXT CONFIGURATION system privilege
- ALTER ANY TRIGGER system privilege
- ALTER ANY VIEW system privilege
- ALTER DATABASE system privilege
- ALTER DATATYPE system privilege
- BACKUP DATABASE system privilege
- CHANGE PASSWORD system privilege
- CHECKPOINT system privilege
- COMMENT ANY OBJECT system privilege
- CREATE ANY INDEX system privilege
- CREATE ANY MATERIALIZED VIEW system privilege
- CREATE ANY OBJECT system privilege
- CREATE ANY PROCEDURE system privilege
- CREATE ANY SEQUENCE system privilege
- CREATE ANY TABLE system privilege
- CREATE ANY TEXT CONFIGURATION system privilege
- CREATE ANY TRIGGER system privilege
- CREATE ANY VIEW system privilege
- CREATE DATATYPE system privilege
- CREATE EXTERNAL REFERENCE system privilege
- CREATE MATERIALIZED VIEW system privilege
- CREATE MESSAGE system privilege
- CREATE PROCEDURE system privilege
- CREATE PROXY TABLE system privilege
- CREATE TABLE system privilege

- CREATE TEXT CONFIGURATION system privilege
- CREATE VIEW system privilege
- DEBUG ANY PROCEDURE system privilege
- DELETE ANY TABLE system privilege
- DROP ANY INDEX system privilege
- DROP ANY MATERIALIZED VIEW system privilege
- DROP ANY OBJECT system privilege
- DROP ANY PROCEDURE system privilege
- DROP ANY SEQUENCE system privilege
- DROP ANY TABLE system privilege
- DROP ANY TEXT CONFIGURATION system privilege
- DROP ANY VIEW system privilege
- DROP CONNECTION system privilege
- DROP DATATYPE system privilege
- DROP MESSAGE system privilege
- EXECUTE ANY PROCEDURE system privilege
- LOAD ANY TABLE system privilege
- INSERT ANY TABLE system privilege
- MANAGE ANY DBSPACE system privilege
- MANAGE ANY EVENT system privilege
- MANAGE ANY EXTERNAL ENVIRONMENT system privilege
- MANAGE ANY EXTERNAL OBJECT system privilege
- MANAGE ANY LDAP SERVER system privilege
- MANAGE ANY LOGIN POLICY system privilege
- MANAGE ANY MIRROR SERVER system privilege
- MANAGE ANY OBJECT PRIVILEGES system privilege
- MANAGE ANY SPATIAL OBJECT system privilege
- MANAGE ANY STATISTICS system privilege
- MANAGE ANY USER system privilege
- MANAGE ANY WEB SERVICE system privilege
- MANAGE AUDITING system privilege
- MANAGE MULTIPLEX system privilege
- MANAGE PROFILING system privilege
- MANAGE REPLICATION system privilege
- MANAGE ROLES system privilege
- MONITOR system privilege
- READ CLIENT FILE system privilege
- READ FILE system privilege
- REORGANIZE ANY OBJECT system privilege

SQL Statements

- SELECT ANY TABLE system privilege
- SERVER OPERATOR system privilege
- SET ANY PUBLIC OPTION system privilege
- SET ANY SECURITY OPTION system privilege
- SET ANY SYSTEM OPTION system privilege
- SET ANY USER DEFINED OPTION system privilege
- SET USER system privilege (granted with ADMIN ONLY clause)
- TRUNCATE ANY TABLE system privilege
- UPDATE ANY TABLE system privilege
- UPGRADE ROLE system privilege
- USE ANY SEQUENCE system privilege
- VALIDATE ANY OBJECT system privilege
- WRITE CLIENT FILE system privilege
- WRITE FILE system privilege

REVOKE USAGE ON SEQUENCE Statement

Removes USAGE privilege on a specified sequence.

Syntax

```
REVOKE USAGE ON SEQUENCE sequence-name  
FROM userID [, ...]
```

Parameters

- **userID** – must be the name of an existing user or role that has a login password. Separate multiple userIDs with commas.

Standards

- SQL – syntax is a Persistent Stored Module feature.
- Sybase – the security model is different in Adaptive Server Enterprise and SAP Sybase IQ, so other syntaxes differ.

Permissions

Requires one of:

- MANAGE ANY OBJECT PRIVILEGE system privilege.
- You own the sequence.

ROLLBACK Statement

Undoes any changes made since the last **COMMIT** or **ROLLBACK**.

Syntax

```
ROLLBACK [ WORK ]
```

Usage

ROLLBACK ends a logical unit of work (transaction) and undoes all changes made to the database during this transaction. A transaction is the database work done between **COMMIT** or **ROLLBACK** statements on one database connection.

Side Effects

- Closes all cursors not opened **WITH HOLD**.
- Releases locks held by the transaction issuing the **ROLLBACK**.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

None, but user must be connected to the database.

See also

- *COMMIT Statement* on page 102
- *ROLLBACK TO SAVEPOINT Statement* on page 413

ROLLBACK TO SAVEPOINT Statement

Cancels any changes made since a savepoint was established. Changes made prior to the savepoint are not undone; they are still pending.

Syntax

```
ROLLBACK TO SAVEPOINT [ savepoint-name ]
```

Parameters

- **savepoint-name** – an identifier that was specified on a **SAVEPOINT** statement within the current transaction. If *savepoint-name* is omitted, the most recent savepoint is used. Any savepoints since the named savepoint are automatically released.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Savepoints are not supported by Adaptive Server Enterprise. To implement similar features in an Adaptive Server Enterprise-compatible manner, you can use nested transactions.

Permissions

None, but there must have been a corresponding **SAVEPOINT** within the current transaction.

See also

- *RELEASE SAVEPOINT Statement* on page 386
- *ROLLBACK Statement* on page 413
- *SAVEPOINT Statement* on page 415

ROLLBACK TRANSACTION Statement [T-SQL]

Cancels any changes made since a savepoint was established using **SAVE TRANSACTION**. Changes made prior to the **SAVE TRANSACTION** are not undone; they are still pending.

Syntax

```
ROLLBACK TRANSACTION [ savepoint-name ]
```

Parameters

- **savepoint-name** – an identifier that was specified on a **SAVE TRANSACTION** statement within the current transaction. If *savepoint-name* is omitted, all outstanding changes are rolled back. Any savepoints since the named savepoint are automatically released.

Examples

- **Example 1** – returns five rows with values 10, 20, and so on. The effect of the delete, but not the prior inserts or update, is undone by the **ROLLBACK TRANSACTION** statement:

```
BEGIN
SELECT row_num INTO #tmp
FROM sa_rowgenerator( 1, 5 )
```



```

UPDATE #tmp SET row_num=row_num*10
SAVE TRANSACTION before_delete
DELETE FROM #tmp WHERE row_num >= 3
ROLLBACK TRANSACTION before_delete
SELECT * FROM #tmp
END

```

Standards

- Vendor extension to ISO/ANSI SQL grammar.

Permissions

None, but there must be a corresponding **SAVE TRANSACTION** within the current transaction.

See also

- *BEGIN TRANSACTION Statement [T-SQL]* on page 88
- *SAVE TRANSACTION Statement [T-SQL]* on page 416

SAVEPOINT Statement

Establishes a savepoint within the current transaction.

Syntax

```
SAVEPOINT [ savepoint-name ]
```

Parameters

- **savepoint-name** – an identifier that can be used in a **RELEASE SAVEPOINT** or **ROLLBACK TO SAVEPOINT** statement.

Usage

All savepoints are automatically released when a transaction ends.

Savepoints that are established while a trigger is executing or while an atomic compound statement is executing are automatically released when the atomic operation ends.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.

- Sybase—Not supported in Adaptive Server Enterprise. To implement similar features in an Adaptive Server Enterprise-compatible manner, use nested transactions.

Permissions

None

See also

- *RELEASE SAVEPOINT Statement* on page 386
- *ROLLBACK TO SAVEPOINT Statement* on page 413

SAVE TRANSACTION Statement [T-SQL]

Establishes a savepoint within the current transaction.

Syntax

```
SAVE TRANSACTION [ savepoint-name ]
```

Parameters

- **savepoint-name** – an identifier that can be used in a **ROLLBACK TRANSACTION** statement. All savepoints are automatically released when a transaction ends.

Examples

- **Example 1** – returns five rows with values 10, 20, and so on. The effect of the delete, but not the prior inserts or update, is undone by the **ROLLBACK TRANSACTION** statement:

```
BEGIN
  SELECT row_num INTO #tmp
  FROM sa_rowgenerator( 1, 5 )
  UPDATE #tmp SET row_num=row_num*10
  SAVE TRANSACTION before_delete
  DELETE FROM #tmp WHERE row_num >= 3
  ROLLBACK TRANSACTION before_delete
  SELECT * FROM #tmp
END
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.

Permissions

None

See also

- *BEGIN TRANSACTION Statement [T-SQL]* on page 88
- *ROLLBACK TRANSACTION Statement [T-SQL]* on page 414

SELECT Statement

Retrieves information from the database.

Syntax

```

SELECT [ ALL | DISTINCT ] [ FIRST | TOP number-of-rows ] select-list
... [ INTO { host-variable-list | variable-list | table-name } ]
... [ INTO LOCAL TEMPORARY TABLE { table-name } ]
... [ FROM table-list ]
... [ WHERE search-condition ]
... [ GROUP BY [ expression [, ...]
                | ROLLUP ( expression [, ...] )
                | CUBE ( expression [, ...] ) ] ]
... [ HAVING search-condition ]
... [ ORDER BY { expression | integer } [ ASC | DESC ] [, ...] ]
| [ FOR JSON json-mode ]
... [ row-limitation-option ]

```

select-list:

```

{ column-name
| expression [ [ AS ] alias-name ]
| * }

```

row-limitation-option:

```

LIMIT { [ offset-expression, ] limit-expression
| limit-expression OFFSET offset-expression }

```

limit-expression:

```
simple-expression
```

offset-expression:

```
simple-expression
```

simple-expression:

```

integer
| variable
| ( simple-expression )
| ( simple-expression { + | - | * } simple-expression )

```

Parameters

- **ALL** or **DISTINCT** – filters query results. If neither is specified, all rows that satisfy the clauses of the **SELECT** statement are retrieved. If **DISTINCT** is specified, duplicate output rows are eliminated. This is called the projection of the result of the statement. In many

cases, statements take significantly longer to execute when **DISTINCT** is specified, so reserve the use of **DISTINCT** for cases where it is necessary.

If **DISTINCT** is used, the statement cannot contain an aggregate function with a **DISTINCT** parameter.

- **FIRST** or **TOP *number-of-rows*** – specifies the number of rows returned from a query. **FIRST** returns the first row selected from the query. **TOP** returns the specified number of rows from the query where *number-of-rows* is in the range 1 – 2147483647 and can be an integer constant or integer variable.

FIRST and **TOP** are used primarily with the **ORDER BY** clause. If you use these keywords without an **ORDER BY** clause, the result might vary from run to run of the same query, as the optimizer might choose a different query plan.

FIRST and **TOP** are permitted only in the top-level **SELECT** of a query, so they cannot be used in derived tables or view definitions. Using **FIRST** or **TOP** in a view definition might result in the keyword being ignored when a query is run on the view.

Using **FIRST** is the same as setting the `ROW_COUNT` database option to 1. Using **TOP** is the same as setting the `ROW_COUNT` option to the same number of rows. If both **TOP** and `ROW_COUNT` are set, then the value of **TOP** takes precedence.

The `ROW_COUNT` option could produce inconsistent results when used in a query involving global variables, system functions or proxy tables. See *ROW_COUNT Option* for details.

- ***select-list*** – is a comma delimited list of expressions that specify what is retrieved from the database. If an asterisk (*) is specified, all columns of all tables in the **FROM** clause (`table-name` all columns of the named table) are selected. Aggregate functions and analytical functions are allowed in the *select-list*.

Note: In SAP Sybase IQ, scalar subqueries (nested selects) are allowed in the select list of the top level **SELECT**, as in SQL Anywhere and Adaptive Server Enterprise. Subqueries cannot be used inside a conditional value expression (for example, in a **CASE** statement).

Subqueries can also be used in a **WHERE** or **HAVING** clause predicate (one of the supported predicate types). However, inside the **WHERE** or **HAVING** clause, subqueries cannot be used inside a value expression or inside a **CONTAINS** or **LIKE** predicate. Subqueries are not allowed in the **ON** clause of outer joins or in the **GROUP BY** clause.

-
- ***alias-names*** – can be used throughout the query to represent the aliased expression. Alias names are also displayed by Interactive SQL at the top of each column of output from the **SELECT** statement. If the optional *alias-name* is not specified after an expression, Interactive SQL displays the expression. If you use the same name or expression for a column alias as the column name, the name is processed as an aliased column, not a table column name.
 - ***INTO host-variable-list*** – specifies where the results of the **SELECT** statement goes. There must be one *host-variable* item for each item in the *select-list*. Select list items are put into the host variables in order. An indicator host variable is also allowed with each

host-variable so the program can tell if the select list item was NULL. Used in Embedded SQL only.

- **INTO *variable-list*** – specifies where the results of the **SELECT** statement go. There must be one variable for each item in the select list. Select list items are put into the variables in order. Used in procedures only
- **INTO *table-name*** – creates a table and fills the table with data.

If the table name starts with #, the table is created as a temporary table. Otherwise, the table is created as a permanent base table. For permanent tables to be created, the query must satisfy these conditions:

- The *select-list* contains more than one item, and the **INTO** target is a single *table-name* identifier, or
- The select-list contains a * and the **INTO** target is specified as *owner.table*.

To create a permanent table with one column, the table name must be specified as *owner.table*. Omit the owner specification for a temporary table.

This statement causes a **COMMIT** before execution as a side effect of creating the table. Requires the CREATE TABLE system privilege to execute this statement. No permissions are granted on the new table: the statement is a short form for **CREATE TABLE** followed by **INSERT... SELECT**.

A **SELECT INTO** from a stored procedure or function is not permitted, as **SELECT INTO** is an atomic statement and you cannot do **COMMIT**, **ROLLBACK**, or some **ROLLBACK TO SAVEPOINT** statements in an atomic statement.

Tables created using this statement do not have a primary key defined. You can add a primary key using **ALTER TABLE**. A primary key should be added before applying any updates or deletes to the table; otherwise, these operations result in all column values being logged in the transaction log for the affected rows.

Use of this clause is restricted to valid SQL Anywhere queries. SAP Sybase IQ extensions are not supported.

- **INTO LOCAL TEMPORARY TABLE** – creates a local, temporary table and populates it with the results of the query. When you use this clause, you do not need to start the temporary table name with #.
- **FROM *table-list*** – retrieves rows and views specified in the *table-list*. Joins can be specified using join operators. For more information, see *FROM Clause*. A **SELECT** statement with no **FROM** clause can be used to display the values of expressions not derived from tables. For example:

```
SELECT @@version
```

displays the value of the global variable @@version. This is equivalent to:

```
SELECT @@version
FROM DUMMY
```

Note: If you omit the **FROM** clause, or if all tables in the query are in the SYSTEM dbspace, the query is processed by SQL Anywhere instead of SAP Sybase IQ and might behave

differently, especially with respect to syntactic and semantic restrictions and the effects of option settings.

If you have a query that does not require a **FROM** clause, you can force the query to be processed by SAP Sybase IQ by adding the clause “FROM iq_dummy,” where iq_dummy is a one-row, one-column table that you create in your database.

-
- **WHERE** *search-condition* – specifies which rows are selected from the tables named in the **FROM** clause. It is also used to do joins between multiple tables. This is accomplished by putting a condition in the **WHERE** clause that relates a column or group of columns from one table with a column or group of columns from another table. Both tables must be listed in the **FROM** clause.

The use of the same **CASE** statement is not allowed in both the **SELECT** and the **WHERE** clause of a grouped query.

SAP Sybase IQ also supports the disjunction of subquery predicates. Each subquery can appear within the **WHERE** or **HAVING** clause with other predicates and can be combined using the AND or OR operators.

- **GROUP BY** – groups columns, alias names, or functions. **GROUP BY** expressions must also appear in the select list. The result of the query contains one row for each distinct set of values in the named columns, aliases, or functions. The resulting rows are often referred to as groups since there is one row in the result for each group of rows from the table list. In the case of **GROUP BY**, all NULL values are treated as identical. Aggregate functions can then be applied to these groups to get meaningful results.

GROUP BY must contain more than a single constant. You do not need to add constants to the **GROUP BY** clause to select the constants in grouped queries. If the **GROUP BY** expression contains only a single constant, an error is returned and the query is rejected.

When **GROUP BY** is used, the select list, **HAVING** clause, and **ORDER BY** clause cannot reference any identifiers except those named in the **GROUP BY** clause. This exception applies: The *select-list* and **HAVING** clause may contain aggregate functions.

- **ROLLUP operator** – subtotals **GROUP BY** expressions that roll up from a detailed level to a grand total.

The **ROLLUP** operator requires an ordered list of grouping expressions to be supplied as arguments. **ROLLUP** first calculates the standard aggregate values specified in the **GROUP BY**. Then **ROLLUP** moves from right to left through the list of grouping columns and creates progressively higher-level subtotals. A grand total is created at the end. If n is the number of grouping columns, **ROLLUP** creates $n+1$ levels of subtotals.

Restrictions on the **ROLLUP** operator:

- **ROLLUP** supports all of the aggregate functions available to the **GROUP BY** clause, but **ROLLUP** does not currently support **COUNT DISTINCT** and **SUM DISTINCT**.
- **ROLLUP** can be used only in the **SELECT** statement; you cannot use **ROLLUP** in a **SELECT** subquery.

- A multiple grouping specification that combines **ROLLUP**, **CUBE**, and **GROUP BY** columns in the same **GROUP BY** clause is not currently supported.
- Constant expressions as **GROUP BY** keys are not supported.

GROUPING is used with the **ROLLUP** operator to distinguish between stored NULL values and NULL values in query results created by **ROLLUP**.

ROLLUP syntax:

```
SELECT ... [ GROUPING ( column-name ) ...] ...
GROUP BY [ expression [, ...]
| ROLLUP ( expression [, ...] ) ]
```

GROUPING takes a column name as a parameter and returns a Boolean value:

Table 9. Values Returned by GROUPING with the ROLLUP Operator

If the value of the result is	GROUPING returns
NULL created by a ROLLUP operation	1 (TRUE)
NULL indicating the row is a subtotal	1 (TRUE)
not created by a ROLLUP operation	0 (FALSE)
a stored NULL	0 (FALSE)

- **CUBE operator** – analyzes data by forming the data into groups in more than one dimension. **CUBE** requires an ordered list of grouping expressions (dimensions) as arguments and enables the **SELECT** statement to calculate subtotals for all possible combinations of the group of dimensions. The **CUBE** operator is part of the **GROUP BY** clause.

Restrictions on the **CUBE** operator:

- **CUBE** supports all of the aggregate functions available to the **GROUP BY** clause, but **CUBE** does not currently support **COUNT DISTINCT** or **SUM DISTINCT**.
- **CUBE** does not currently support the inverse distribution analytical functions **PERCENTILE_CONT** and **PERCENTILE_DISC**.
- **CUBE** can be used only in the **SELECT** statement; you cannot use **CUBE** in a **SELECT** subquery.
- A multiple **GROUPING** specification that combines **ROLLUP**, **CUBE**, and **GROUP BY** columns in the same **GROUP BY** clause is not currently supported.
- Constant expressions as **GROUP BY** keys are not supported.

GROUPING is used with the **CUBE** operator to distinguish between stored NULL values and NULL values in query results created by **CUBE**.

CUBE syntax:

```
SELECT ... [ GROUPING ( column-name ) ...] ...
GROUP BY [ expression [, ...]
| CUBE ( expression [, ...] ) ]
```

GROUPING takes a column name as a parameter and returns a Boolean value:

Table 10. Values Returned by GROUPING with the CUBE Operator

If the value of the result is	GROUPING returns
NULL created by a CUBE operation	1 (TRUE)
NULL indicating the row is a subtotal	1 (TRUE)
not created by a CUBE operation	0 (FALSE)
a stored NULL	0 (FALSE)

When generating a query plan, the SAP Sybase IQ optimizer estimates the total number of groups generated by the **GROUP BY CUBE** hash operation. The `MAX_CUBE_RESULTS` database option sets an upper boundary for the number of estimated rows the optimizer considers for a hash algorithm that can be run. If the actual number of rows exceeds the `MAX_CUBE_RESULT` option value, the optimizer stops processing the query and returns the error message “Estimate number: nnn exceed the `DEFAULT_MAX_CUBE_RESULT` of `GROUP BY CUBE` or `ROLLUP`”, where *nnn* is the number estimated by the optimizer. See *MAX_CUBE_RESULT Option* for information on setting the `MAX_CUBE_RESULT` option.

- **HAVING search-condition** – based on the group values and not on the individual row values. The **HAVING** clause can be used only if either the statement has a **GROUP BY** clause or if the select list consists solely of aggregate functions. Any column names referenced in the **HAVING** clause must either be in the **GROUP BY** clause or be used as a parameter to an aggregate function in the **HAVING** clause.
- **ORDER BY** – orders the results of a query. Each item in the **ORDER BY** list can be labeled as `ASC` for ascending order or `DESC` for descending order. Ascending is assumed if neither is specified. If the expression is an integer *n*, then the query results are sorted by the *n*th item in the select list.

In Embedded SQL, the **SELECT** statement is used for retrieving results from the database and placing the values into host variables with the **INTO** clause. The **SELECT** statement must return only one row. For multiple row queries, you must use cursors.

You cannot include a Java class in the **SELECT** list, but you can, for example, create a function or variable that acts as a wrapper for the Java class and then select it.

FOR JSON clause specifies that the result set is to be returned in JSON format. The JSON format depends on the mode you specify. This clause cannot be used with the `FOR UPDATE` or `FOR READ ONLY` clause. Cursors declared with `FOR JSON` are implicitly `READ ONLY`.

When you specify `RAW` mode, each row in the result set is returned as a flattened JSON representation.

`AUTO` mode returns the query results as nested JSON objects based on query joins.

EXPLICIT mode allows you to control the form of the generated JSON objects. Using EXPLICIT mode offers more flexibility in specifying columns and nested hierarchical objects to produce uniform or heterogeneous arrays.

- **row-limitation clause** – returns a subset of rows that satisfy the **WHERE** clause. Only one *row-limitation* clause can be specified at a time. When specifying this clause, an **ORDER BY** clause is required to order the rows in a meaningful manner.

The **LIMIT** and **OFFSET** arguments can be simple arithmetic expressions over host variables, integer constants, or integer variables. The **LIMIT** argument must evaluate to a value greater than or equal to 0. The **OFFSET** argument must evaluate to a value greater than or equal to 0. If *offset-expression* is not specified, the default is 0.

The row limitation clause **LIMIT** *offset-expression, limit-expression* is equivalent to **LIMIT** *limit-expression* **OFFSET** *offset-expression*.

The **LIMIT** keyword is disabled by default. Use the `RESERVED_KEYWORDS` option to enable the **LIMIT** keyword.

Examples

- **Example 1** – list all tables and views in the system catalog:

```
SELECT tname
FROM SYS.SYSCATALOG
WHERE tname LIKE 'SYS%' ;
```

- **Example 2** – list all customers and the total value of their orders:

```
SELECT CompanyName,
       CAST( sum(SalesOrderItems.Quantity *
                Products.UnitPrice) AS INTEGER) VALUE
FROM Customers
   LEFT OUTER JOIN SalesOrders
   LEFT OUTER JOIN SalesOrderItems
   LEFT OUTER JOIN Products
GROUP BY CompanyName
ORDER BY VALUE DESC
```

- **Example 3** – list the number of employees:

```
SELECT count(*)
FROM Employees;
```

- **Example 4** – an Embedded SQL SELECT statement:

```
SELECT count(*) INTO :size FROM Employees;
```

- **Example 5** – list the total sales by year, model, and color:

```
SELECT year, model, color, sum(sales)
FROM sales_tab
GROUP BY ROLLUP (year, model, color);
```

- **Example 6** – select all items with a certain discount into a temporary table:

SQL Statements

```
SELECT * INTO #TableTemp FROM lineitem
WHERE l_discount < 0.5
```

- **Example 7** – return information about the employee that appears first when employees are sorted by last name:

```
SELECT FIRST *
FROM Employees
ORDER BY Surname;
```

- **Example 8** – return the first five employees when their names are sorted by last name:

```
SELECT TOP 5 *
FROM Employees
ORDER BY Surname;
```

```
SELECT *
FROM Employees
ORDER BY Surname
LIMIT 5;
```

- **Example 9** – list the fifth and sixth employees sorted in descending order by last name:

```
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT 4,2;
```

- **Example 10** – return the first five employees when their names are sorted by last name:

```
CREATE OR REPLACE VARIABLE atop INT = 10;
```

```
SELECT TOP (atop -5) *
FROM Employees
ORDER BY Surname;
```

```
SELECT *
FROM Employees
ORDER BY Surname
LIMIT (atop-5);
```

- **Example 11** – list the fifth and sixth employees sorted in descending order by last name:

```
CREATE OR REPLACE VARIABLE atop INT = 10;
```

```
SELECT *
FROM Employees
ORDER BY Surname DESC
LIMIT (atop - 8) OFFSET (atop -2 -3 -1);
```

Usage

You can use a **SELECT** statement in Interactive SQL to browse data in the database or to export data from the database to an external file.

You can also use a **SELECT** statement in procedures or in Embedded SQL. The **SELECT** statement with an **INTO** clause is used for retrieving results from the database when the **SELECT** statement returns only one row. (Tables created with **SELECT INTO** do not inherit

IDENTITY/AUTOINCREMENT tables.) For multiple-row queries, you must use cursors. When you select more than one column and do not use *#table*, **SELECT INTO** creates a permanent base table. **SELECT INTO #table** always creates a temporary table regardless of the number of columns. **SELECT INTO** table with a single column selects into a host variable.

Note: When writing scripts and stored procedures that **SELECT INTO** a temporary table, wrap any select list item that is not a base column in a **CAST** expression. This guarantees that the column data type of the temporary table is the required data type.

Tables with the same name but different owners require aliases. A query without aliases returns incorrect results:

```
SELECT * FROM user1.t1
WHERE NOT EXISTS
(SELECT *
FROM user2.t1
WHERE user2.t1.col1 = user1.t.col1);
```

For correct results, use an alias for each table:

```
SELECT * FROM user1.t1 U1
WHERE NOT EXISTS
(SELECT *
FROM user2.t1 U2
WHERE U2.col1 = U1.col1);
```

The **INTO** clause with a *variable-list* is used only in procedures.

In **SELECT** statements, a stored procedure call can appear anywhere a base table or view is allowed. Note that CIS functional compensation performance considerations apply. For example, a **SELECT** statement can also return a result set from a procedure.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by SAP Sybase IQ, with some differences in syntax.

Permissions

Requires SELECT privilege on the named tables and views.

See also

- *CREATE VIEW Statement* on page 234
- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *FETCH Statement [ESQL] [SP]* on page 283
- *FROM Clause* on page 295
- *MAX_CUBE_RESULT Option* on page 576
- *OPEN Statement [ESQL] [SP]* on page 369
- *UNION Operation* on page 446

- *RESERVED_KEYWORDS Option* on page 618
- *ROW_COUNT Option* on page 621
- *SUBQUERY_CACHING_PREFERENCE Option* on page 633

SET Statement [ESQL]

Assigns a value to a SQL variable.

Syntax

```
SET identifier = expression
```

Examples

- **Example 1** – this code fragment inserts a large text value into the database:

```
EXEC SQL BEGIN DECLARE SECTION;
char buffer[5001];
EXEC SQL END DECLARE SECTION;

EXEC SQL CREATE VARIABLE hold_text VARCHAR;
EXEC SQL SET hold_text = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( buffer, 1, 5000, fp );
    if( size <= 0 ) break;

    /* buffer must be null-terminated */
    buffer[size] = '\0';
    /* add data to blob using concatenation */
    EXEC SQL SET hold_text = hold_text || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_text );
EXEC SQL DROP VARIABLE hold_text;
```

- **Example 2** – this code fragment inserts a large binary value into the database:

```
EXEC SQL BEGIN DECLARE SECTION;
DECL_BINARY( 5000 ) buffer;
EXEC SQL END DECLARE SECTION;
EXEC SQL CREATE VARIABLE hold_blob LONG BINARY;
EXEC SQL SET hold_blob = '';
for(;;) {
    /* read some data into buffer ... */
    size = fread( &(buffer.array), 1, 5000, fp );
    if( size <= 0 ) break;
    buffer.len = size;

    /* add data to blob using concatenation
       Note that concatenation works for
       binary data too! */
```

```
EXEC SQL SET hold_blob = hold_blob || :buffer;
}
EXEC SQL INSERT INTO some_table VALUES ( 1, hold_blob );
EXEC SQL DROP VARIABLE hold_blob;
```

Usage

The **SET** statement assigns a new value to a variable that was previously created using the **CREATE VARIABLE** statement.

You can use a variable in a SQL statement anywhere a column name is allowed. If there is no column name that matches the identifier, the database server checks to see if there is a variable that matches, and uses its value.

Variables are local to the current connection, and disappear when you disconnect from the database or when you use **DROP VARIABLE**. They are not affected by **COMMIT** or **ROLLBACK** statements.

Variables are necessary for creating large text or binary objects for **INSERT** or **UPDATE** statements from Embedded SQL programs because Embedded SQL host variables are limited to 32,767 bytes.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Not supported. In Adaptive Server Enterprise, variables are assigned using the **SELECT** statement with no table, a Transact-SQL syntax that is also supported by SAP Sybase IQ. The **SET** statement is used to set database options in Adaptive Server Enterprise.

Permissions

None

See also

- *CREATE VARIABLE Statement* on page 232
- *DROP VARIABLE Statement* on page 276

SET Statement [T-SQL]

Sets database options in an Adaptive Server Enterprise-compatible manner.

Syntax

```
SET option-name option-value
option-value:
```

```

ANSINULL [ ON | OFF ]
| ANSI_PERMISSIONS [ ON | OFF ]
| CLOSE_ON_ENDTRANS ON
| QUOTED_IDENTIFIER [ ON | OFF ]
| ROWCOUNT integer
| STRING_RTRUNCATION [ ON | OFF ]
| TRANSACTION ISOLATION LEVEL [ 0 | 1 | 2 | 3 ]

```

Parameters

- **ANSINULL** – the default behavior for comparing values to NULL in SAP Sybase IQ and Adaptive Server Enterprise is different. Setting ANSINULL to OFF provides Transact-SQL compatible comparisons with NULL.
- **ANSI_PERMISSIONS** – the default behavior in SAP Sybase IQ and Adaptive Server Enterprise regarding permissions required to carry out a DELETE containing a column reference is different. Setting ANSI_PERMISSIONS to OFF provides Transact-SQL-compatible permissions on DELETE.
- **CLOSE_ON_ENDTRANS** – when set to ON (the default and only allowable value), cursors are closed at the end of a transaction. With the option set ON, CLOSE_ON_ENDTRANS provides Transact-SQL-compatible behavior.
- **QUOTED_IDENTIFIER** – controls whether strings enclosed in double quotes are interpreted as identifiers (ON) or as literal strings (OFF).
- **ROWCOUNT** – in the Transact-SQL, limits to the specified integer the number of rows fetched for any cursor. This includes rows fetched by repositioning the cursor. Any fetches beyond this maximum return a warning. The setting is considered when returning the estimate of the number of rows for a cursor on an **OPEN** request.

Note: SAP Sybase IQ supports the *@@rowcount* global variable. **SELECT**, **INSERT**, **DELETE**, and **UPDATE** statements affect the value of the ROWCOUNT clause. The **ROWCOUNT** clause has no effect on cursor operation, the **IF** statement, or creating or dropping a table or procedure.

In SAP Sybase IQ, if ROWCOUNT is greater than the number of rows that **dbisql** can display, **dbisql** may do extra fetches to reposition the cursor. The number of rows actually displayed may be less than the number requested. Also, if any rows are refetched due to truncation warnings, the count might be inaccurate.

A value of zero resets the option to get all rows.

- **STRING_RTRUNCATION** – the default behavior in SAP Sybase IQ and Adaptive Server Enterprise when nonspace characters are truncated on assigning SQL string data is different. Setting STRING_RTRUNCATION to ON provides Transact-SQL-compatible string comparisons, including hexadecimal string (binary data type) comparisons.

- **TRANSACTION ISOLATION LEVEL** – sets the locking isolation level for the current connection. For Adaptive Server Enterprise, only 1 and 3 are valid options. For SAP Sybase IQ, only 3 is a valid option.
- **SET PREFETCH** – is allowed by SAP Sybase IQ for compatibility, but has no effect.

Usage

Database options in SAP Sybase IQ are set using the **SET OPTION** statement. However, SAP Sybase IQ also provides support for the Adaptive Server Enterprise **SET** statement for a set of options particularly useful for compatibility.

Standards

- SQL—Transact-SQL extension to ISO/ANSI SQL grammar.
- Sybase—SAP Sybase IQ supports a subset of the Adaptive Server Enterprise database options.

Permissions

None

See also

- *SET OPTION Statement* on page 431

SET CONNECTION Statement [ESQL] [Interactive SQL]

Changes the active database connection.

Syntax

```
SET CONNECTION [connection-name]
```

Parameters

- **connection-name** – identifier, string, or host-variable

Examples

- **Example 1** – in Embedded SQL:

```
EXEC SQL SET CONNECTION :conn_name
```

- **Example 2** – from **dbisql**, set the current connection to the connection named “conn1”:

```
SET CONNECTION conn1
```

Usage

The current connection state is saved and is resumed when it again becomes the active connection. If *connection-name* is omitted and there is a connection that was not named, that connection becomes the active connection.

Note: When cursors are opened in Embedded SQL, they are associated with the current connection. When the connection is changed, the cursor names are not accessible. The cursors remain active and in position and become accessible when the associated connection becomes active again.

Standards

- SQL—**dbisql** use is a vendor extension to ISO/ANSI SQL grammar. Embedded SQL is a full-level feature.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *CONNECT Statement [ESQL] [Interactive SQL]* on page 104
- *DISCONNECT Statement [Interactive SQL]* on page 254

SET DESCRIPTOR Statement [ESQL]

Describes the variables in a SQL descriptor area, and places data into the descriptor area.

Syntax

```
SET DESCRIPTOR descriptor-name
... { COUNT = { integer | hostvar }
    | VALUE n assignment [, ...] }

assignment:
{ { TYPE
  | SCALE
  | PRECISION
  | LENGTH
  | INDICATOR } = { integer
  | hostvar }
  | DATA = hostvar }
```


Parameters

- **COUNT** – sets the number of described variables within the descriptor area. The value for count cannot exceed the number of variables specified when the descriptor area was allocated.
- **VALUE** – the value *n* specifies the variable in the descriptor area upon which the assignments are performed.
- **DATA** – type checking is performed when using the DATA clause to ensure that the variable in the descriptor area has the same type as the host variable. If an error occurs, the code is returned in the SQLCA.

Examples

- **Example 1** – see *ALLOCATE DESCRIPTOR Statement [ESQL]*.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

See also

- *ALLOCATE DESCRIPTOR Statement [ESQL]* on page 5
- *DEALLOCATE DESCRIPTOR Statement [ESQL]* on page 237

SET OPTION Statement

Changes options that affect the behavior of the database and its compatibility with Transact-SQL. Setting the value of an option can change the behavior for all users or an individual user, in either a temporary or permanent scope.

Syntax

```
SET [ EXISTING ] [ TEMPORARY ] OPTION
... [ userid. | PUBLIC.] option-name = [ option-value ]
```

Parameters

- **option-value** – a host-variable (indicator allowed), string, identifier, or number. The maximum length of *option-value* when set to a string is 127 bytes.

If *option-value* is omitted, the specified option setting is deleted from the database. If it was a personal option setting, the value used reverts to the PUBLIC setting.

Note: For all database options that accept integer values, SAP Sybase IQ truncates any decimal *option-value* setting to an integer value. For example, the value 3.8 is truncated to 3.

- **EXISTING** – option values cannot be set for an individual user ID unless there is already a PUBLIC user ID setting for that option.
- **TEMPORARY** – changes the duration that the change takes effect. Without the TEMPORARY clause, an option change is permanent: it does not change until it is explicitly changed using **SET OPTION**.

When the TEMPORARY clause is applied using an individual user ID, the new option value is in effect as long as that user is logged in to the database.

When the TEMPORARY clause is used with the PUBLIC user ID, the change is in place for as long as the database is running. When the database is shut down, TEMPORARY options for the PUBLIC user ID revert to their permanent value.

If a TEMPORARY option is deleted, the option setting reverts to the permanent setting.

Examples

- **Example 1** – set the DATE_FORMAT option:

```
SET OPTION public.date_format = 'Mmm dd yyyy'
```

- **Example 2** – set the WAIT_FOR_COMMIT option to on:

```
SET OPTION wait_for_commit = 'on'
```

- **Example 3** – embedded SQL examples:

```
EXEC SQL SET OPTION :user.:option_name = :value;  
EXEC SQL SET TEMPORARY OPTION Date_format = 'mm/dd/yyyy';
```

Usage

The classes of options are:

- General database options
- Transact-SQL compatibility database options

Specifying either a user ID or the PUBLIC user ID determines whether the option is set for an individual user, a role represented by *userid*, or the PUBLIC user ID (the role to which all users are a member). If the option applies to a role ID, option settings are not inherited by members of the role—the change is applied only to the role ID. If no role is specified, the option change is applied to the currently logged-in user ID that issued the **SET OPTION** statement. For example, this statement applies an option change to the PUBLIC user ID:

```
SET OPTION Public.login_mode = standard
```

In Embedded SQL, only database options can be set temporarily.

Changing the value of an option for the PUBLIC user ID sets the value of the option for any user that has not set its own value. Option values cannot be set for an individual user ID unless there is already a PUBLIC user ID setting for that option.

Temporarily setting an option for the PUBLIC user ID, as opposed to setting the value of the option permanently, offers a security advantage. For example, when the **LOGIN_MODE** option is enabled, the database relies on the login security of the system on which it is running.

Enabling the option temporarily means a database relying on the security of a Windows domain is not compromised if the database is shut down and copied to a local machine. In that case, the temporary enabling of **LOGIN_MODE** reverts to its permanent value, which might be Standard, a mode in which integrated logins are not permitted.

Warning! Changing option settings while fetching rows from a cursor is not supported, as it can lead to unpredictable behavior. For example, changing the DATE_FORMAT setting while fetching from a cursor returns different date formats among the rows in the result set. Do not change option settings while fetching rows.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Adaptive Server Enterprise. SAP Sybase IQ does support some Adaptive Server Enterprise options using the **SET** statement.

Permissions

No specific system privileges are required to set your own options.

The SET ANY PUBLIC OPTION system privilege is required to set database options for another user.

The SET ANY SYSTEM OPTION system privilege is required to set a SYSTEM option for the PUBLIC user ID.

The SET ANY SECURITY OPTION system privilege is required to set a SECURITY option for the PUBLIC user ID.

See also

- *Database Options* on page 463

SET OPTION Statement [Interactive SQL]

Changes Interactive SQL (**dbisql**) options.

Syntax

Syntax 1

```
SET [ TEMPORARY ] OPTION
... [ userid. | PUBLIC.]option-name = [ option-value ]
```

Syntax 2

```
SET PERMANENT
```

Syntax 3

```
SET
```

Parameters

- **userid** – identifier, string, or host-variable
- **option-name** – identifier, string, or host-variable
- **option-value** – host-variable (indicator allowed), string, identifier, or number

Usage

SET PERMANENT (Syntax 2) stores all current **dbisql** options in the `SYSOPTION` system table. These settings are automatically established every time **dbisql** is started for the current user ID.

Syntax 3 is used to display all of the current option settings. If there are temporary options set for **dbisql** or the database server, these display; otherwise, permanent option settings are displayed.

If you incorrectly type the name of an option when you are setting the option, the incorrect name is saved in the `SYSOPTION` table. You can remove the incorrectly typed name from the `SYSOPTION` table by setting the option `PUBLIC` with an equality after the option name and no value:

```
SET OPTION PUBLIC.a_mistyped_name=;
```

See also

- *Database Options* on page 463

SET SQLCA Statement [ESQL]

Tells the SQL preprocessor to use a SQLCA other than the default global *sqlca*.

Syntax

```
SET SQLCA sqlca
```

Parameters

- *sqlca* – identifier or string

Examples

- **Example 1** – this function can be found in a Windows DLL. Each application that uses the DLL has its own SQLCA.

```
an_sql_code FAR PASCAL ExecutesSQL( an_application *app, char
*com )
{
    EXEC SQL BEGIN DECLARE SECTION;
    char *sqlcommand;
    EXEC SQL END DECLARE SECTION;
    EXEC SQL SET SQLCA "&app->.sqlca";
    sqlcommand = com;
    EXEC SQL WHENEVER SQLERROR CONTINUE;
    EXEC SQL EXECUTE IMMEDIATE :sqlcommand;
    return( SQLCODE );
}
```

Usage

The current SQLCA pointer is implicitly passed to the database interface library on every Embedded SQL statement. All Embedded SQL statements that follow this statement in the C source file use the new SQLCA. This statement is necessary only when you are writing code that is reentrant. The *sqlca* should reference a local variable. Any global or module static variable is subject to being modified by another thread.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not supported by Open Client/Open Server.

Permissions

None

SETUSER Statement

Allows a user to temporarily assume the roles and system privileges of another user (also known as impersonation) to perform operations, provided they already have the minimum required privileges to perform the task to begin with.

Note: The SET USER system privilege is two words; the SETUSER statement is one word.

Syntax

```
SETUSER userID
```

Parameters

- **UserID** – must be the name of an existing user or role that has a login password.

Usage

At-least criteria validation occurs when the SETUSER statement is executed, not when the SET USER system privilege is granted.

To terminate a successful impersonation, issue the SETUSER statement without specifying a userID.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

1. The impersonator has been granted the right to impersonate the target user.
2. The impersonator has, at minimum, all the roles and system privileges granted to the target user.
3. The impersonator has been granted the said roles and system privileges with similar or higher administrative rights.

Note: For the purposes of meeting administrative rights criteria, the WITH ADMIN OPTION and WITH ADMIN ONLY OPTION clauses are considered to grant similar administrative rights. They are also considered to grant higher administrative rights than the WITH NO ADMIN OPTION clause. For example, User1 is granted Role1 with the WITH ADMIN OPTION clause, User2 is granted Role1 with the WITH ADMIN ONLY clause, and User3 is granted Role1 with the WITH NO ADMIN OPTION clause. User1 and User2 are said to be granted Role1 with similar administrative rights. User1 and User2 are also said to be granted Role1 with higher administrative rights than User3.

4. If the target user has been granted a system privilege which supports extensions, the clauses used to grant the system privilege to the impersonator are a super-set of those used for the target user. Currently, only the SET USER and CHANGE PASSWORD system privileges support extensions.

Note:

- The ANY clause is considered a super-set of the *target_roles_list* and *target_users_list* clauses. If the target user has been granted the SET USER system privilege with an ANY grant, the impersonator must also have the ANY grant.
 - If the target user has been granted the SET USER system privilege with both the *target_roles_list* and *target_users_list* clauses, the impersonator must also have been granted the system privilege with the two clauses, and the target list of each clause must be equal to or a super-set of the corresponding clause grant of the target user. For example, if the target lists of both the impersonator and target user contain User1, User2 and Role1, Role2, respectively, the target list grants for each clause are said to be equal. Alternately, if the target list grants of the impersonator contain User1, User2, Role1, Role2, respectively, while the target list grants of the target user contain User1, Role2 only, the target list grants of the impersonator are said to be a super-set of the target user.
 - If the target user has been granted the SET USER system privilege with a single target list clause, the target list of the impersonator must be equal to or a super-set of the list of the target user. For example, the target_user_list of both the impersonator and the target user contain User1 and User2 (equal) or the impersonator list contains User1, User2, while the target user contains User2; User1, User2 (impersonator list) is a super-set of User2 (target user list).
 - By definition, a user can always impersonate themselves. Therefore, if the target user has been granted the right to impersonate the impersonator, this does not violate the equal to or a super-set of criteria requirement of the impersonator. For example, User3 is the impersonator and User4 is the target user. The target_user_list for User3 contains User4 and User5. The target_user_list for User4 contains User3 and User5. If you remove the impersonator from the target list, the target list of User3 meets the criteria requirement.
-

SIGNAL Statement

Lets you raise an exception condition.

Syntax

SIGNAL *exception-name*

Standards

- SQL—ISO/ANSI SQL compliant.

- Sybase—**SIGNAL** is not supported by Adaptive Server Enterprise.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84
- *RESIGNAL Statement* on page 388

START DATABASE Statement [Interactive SQL]

Starts a database on the specified database server.

Syntax

```
START DATABASE database-file
... [ AS database-name ]
... [ ON engine-name ]
... [ AUTOSTOP { YES | NO } ]
... [ KEY key ]
```

Parameters

- **database-name** – if not specified, a default name is assigned to the database. This default name is the root of the database file. For example, a database in file `c:\sybase\16_0\demo\iqdemo.db` is given the default name `iqdemo`.
- **engine-name** – if not specified, the default database server is assumed. The default database server is the first started server among those currently running.
- **AUTOSTOP** – when set to YES (default), the database is unloaded when the last connection to it is dropped. When set to NO, the database is not unloaded.
- **KEY** – to enter the KEY value (password) for strongly encrypted databases.

Note: Start only one database on a given SAP Sybase IQ database server.

Examples

- **Example 1** – on a UNIX system, start the database file `/s1/sybase/sample_2.db` on the current server:

```
START DATABASE '/s1/sybase/sample_2.db'
```

- **Example 2** – on a Windows system, start the database file `c:\sybase\sample_2.db` as `sam2` on the server `eng1`:


```
START DATABASE 'c:\sybase\sample_2.db'
AS sam2
ON eng1
```

Usage

The database server must be running. The full path must be specified for the database file unless the file is located in the current directory.

The **START DATABASE** statement does not connect **dbisql** to the specified database: a **CONNECT** statement must be issued to make a connection.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

Requires the SERVER OPERATOR system privilege.

START ENGINE Statement [Interactive SQL]

Starts a database server.

Syntax

```
START ENGINE
  AS engine-name [ STARTLINE command-string ]
```

Parameters

- **STARTLINE** – to specify a set of options for the server.
- **command-string** – valid command strings are those that conform to the database server command line description per the *Utility Guide*, under *start_iq Database Server Startup Utility*.

Examples

- **Example 1** – starts a database server named eng1 without starting any databases on it:

```
START ENGINE AS eng1
```

- **Example 2** – starts the same server with a cache of 8096KB:

```
START ENGINE AS eng1 STARTLINE 'start_iq -c 8096'
```

Usage

Several server options are required for SAP Sybase IQ to operate well. To ensure that you are using the right set of options, start your server by using either Sybase Control Center or a configuration file with the **start_iq** command.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *STOP ENGINE Statement [Interactive SQL]* on page 442

START JAVA Statement

Loads the Java VM at a convenient time, so that when the user starts to use Java functionality, there is no initial pause while the Java VM is loaded.

Syntax

```
START EXTERNAL ENVIRONMENT JAVA
```

Examples

- **Example 1** – start the Java VM:

```
START EXTERNAL ENVIRONMENT JAVA
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *STOP JAVA Statement* on page 442

STOP DATABASE Statement [Interactive SQL]

Stops a database on the specified database server.

Syntax

```
STOP DATABASE database-name
... [ ON engine-name ]
... [ UNCONDITIONALLY ]
```

Parameters

- **database-name** – the name specified in the -n parameter when the database is started, or specified in the DBN (DatabaseName) connection parameter. This name is typically the file name of the database file that holds the catalog store, without the .db extension, but can be any user-defined name.
- **engine-name** – if not specified, all running engines are searched for a database of the specified name.
- **UNCONDITIONALLY** – if specified, the database is stopped, even if there are connections to the database. If not specified, the database is not stopped if there are connections to it.

Examples

- **Example 1** – stop the database named `sample` on the default server:

```
STOP DATABASE sample
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

Requires the SERVER OPERATOR system privilege.

See also

- *DISCONNECT Statement [Interactive SQL]* on page 254
- *START DATABASE Statement [Interactive SQL]* on page 438

STOP ENGINE Statement [Interactive SQL]

Stops a database server.

Syntax

```
STOP ENGINE engine-name [ UNCONDITIONALLY ]
```

Parameters

- **UNCONDITIONALLY** – if specified, the database server is stopped, even if there are connections to the server. If not specified, the database server is not stopped if there are connections to it.

Examples

- **Example 1** – stop the database server named `sample`:

```
STOP ENGINE sample
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Not applicable.

Permissions

None

See also

- *START ENGINE Statement [Interactive SQL]* on page 439

STOP JAVA Statement

Releases resources associated with the Java VM to economize on the use of system resources.

Syntax

```
STOP EXTERNAL ENVIRONMENT JAVA
```

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.

- Sybase—Not applicable.

Permissions

None

See also

- *START JAVA Statement* on page 440

TRIGGER EVENT Statement

Triggers a named event. The event may be defined for event triggers or be a scheduled event.

Syntax

```
TRIGGER EVENT event-name [ ( parm = value, ... ) ]
```

Usage

Actions are tied to particular trigger conditions or schedules by a **CREATE EVENT** statement. You can use **TRIGGER EVENT** to force the event handler to execute, even when the scheduled time or trigger condition has not occurred. **TRIGGER EVENT** does not execute disabled event handlers

When a triggering condition causes an event handler to execute, the database server can provide context information to the event handler using the `event_parameter` function. **TRIGGER EVENT** allows you to explicitly supply these parameters, to simulate a context for the event handler.

When you trigger an event, specify the event name. You can list event names by querying the system table `SYSEVENT`. For example:

```
SELECT event_id, event_name FROM SYS.SYSEVENT
```

Permissions

Requires the `MANAGE ANY EVENT` system privilege.

See also

- *ALTER EVENT Statement* on page 14
- *CREATE EVENT Statement* on page 123

TRUNCATE Statement

Deletes all rows from a table or materialized view without deleting the table definition.

Syntax

Syntax 1

```
TRUNCATE
  TABLE [ owner.]table-name
  | MATERIALIZED VIEW owner.] materialized-view-name
```

Syntax 2

```
TRUNCATE TABLE [ owner .]table
  [ PARTITION partition-name
  | SUBPARTITION subpartition-name ]
```

Parameters

- **PARTITION** – specifies which partition to truncate, and does not affect data in other partitions.

Note: Specifying an RLV-enabled table in the PARTITION clause results in an error.

- **SUBPARTITION** – to truncate tables partitioned by a composite partitioning scheme.

Note: Specifying an RLV-enabled table in the SUBPARTITION clause results in an error.

Examples

- **Example 1** – delete all rows from the Sale table:

```
TRUNCATE TABLE Sale
```

Usage

TRUNCATE is equivalent to a **DELETE** statement without a **WHERE** clause, except that each individual row deletion is not entered into the transaction log. After a **TRUNCATE TABLE** statement, the table structure and all of the indexes continue to exist until you issue a **DROP TABLE** statement. The column definitions and constraints remain intact, and permissions remain in effect.

The **TRUNCATE** statement is entered into the transaction log as a single statement, like data definition statements. Each deleted row is not entered into the transaction log.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

Requires one of:

- TRUNCATE ANY TABLE system privilege.
- ALTER ANY TABLE system privilege.
- ALTER ANY OBJECT system privilege.
- TRUNCATE privilege on the table.
- You own the object.

For both temporary and base tables, you can execute **TRUNCATE TABLE** while other users have read access to the table. This behavior differs from SQL Anywhere, which requires exclusive access to truncate a base table. SAP Sybase IQ table versioning ensures that **TRUNCATE TABLE** can occur while other users have read access; however, the version of the table these users see depends on when the read and write transactions commit.

See also

- *DELETE Statement* on page 248

TRUNCATE TEXT INDEX Statement

Deletes the data in a MANUAL or an AUTO REFRESH text index.

Syntax

```
TRUNCATE TEXT INDEX text-index-name
ON [ owner. ] table-name
```

Parameters

- **ON** – the name of the table on which the text index is built.

Examples

- **Example 1** – the first statement creates the `txt_index_manual` text index. The second statement populates the text index with data. The third statement truncates the text index data.

```
CREATE TEXT INDEX txt_index_manual ON GROUPO.MarketingInformation
( Description )
  MANUAL REFRESH;
REFRESH TEXT INDEX txt_index_manual ON
```

SQL Statements

```
GROUPO.MarketingInformation;  
TRUNCATE TEXT INDEX txt_index_manual ON  
GROUPO.MarketingInformation;
```

The truncated text index is repopulated with data the next time it is refreshed.

Usage

Use the TRUNCATE TEXT INDEX statement when you want to delete data from a manual text index without dropping the text index definition. For example, to alter the text configuration object for the text index to change the stoplist, truncate the text index, change the text configuration object it refers to, and then refresh the text index to populate it with new data.

You cannot perform a TRUNCATE TEXT INDEX statement on a text index defined as IMMEDIATE REFRESH (the default). For IMMEDIATE REFRESH text indexes, you must drop the index instead.

The TRUNCATE TEXT INDEX requires exclusive access to the table. Any open cursors that reference the table being truncated must be closed, and a COMMIT or ROLLBACK statement must be executed to release the reference to the table.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- ALTER ANY INDEX system privilege.
- ALTER ANY OBJECT system privilege.
- REFERENCES privilege on the table.
- You own the table.

UNION Operation

Combines the results of two or more select statements.

Syntax

```
select-without-order-by  
... UNION [ ALL ] select-without-order-by  
... [ UNION [ ALL ] select-without-order-by ]...  
... [ ORDER BY integer [ ASC | DESC ] [, ... ] ]
```


Parameters

- **ALL** – the results of UNION ALL are the combined results of the component **SELECT** statements. The results of UNION are the same as UNION ALL, except that duplicate rows are eliminated. Eliminating duplicates requires extra processing, so UNION ALL should be used instead of UNION where possible.
- **ORDER BY** – only integers are allowed in the order by list. These integers specify the position of the columns to be sorted.

Examples

- **Example 1** – list all distinct surnames of employees and customers:

```
SELECT Surname
FROM Employees
UNION
SELECT Surname
FROM Customers
```

Usage

The results of several **SELECT** statements can be combined into a larger result using a UNION clause. The component **SELECT** statements must each have the same number of items in the select list, and cannot contain an ORDER BY clause. See *FROM Clause*.

If corresponding items in two select lists have different data types, SAP Sybase IQ chooses a data type for the corresponding column in the result, and automatically converts the columns in each component **SELECT** statement appropriately.

The column names displayed are the same column names that display for the first **SELECT** statement.

Note: When **SELECT** statements include constant values and UNION ALL views but omit the FROM clause, use `iq_dummy` to avoid errors. See *FROM Clause* for details.

Standards

- SQL—ISO/ANSI SQL compliant.
- Sybase—Supported by Adaptive Server Enterprise, which also supports a **COMPUTE** clause.

Permissions

Requires SELECT privilege for each component of the **SELECT** statements.

See also

- *FROM Clause* on page 295
- *SELECT Statement* on page 417

UPDATE Statement

Modifies existing rows of a single table, or a view that contains only one table.

Syntax

```
UPDATE table
... SET [column-name = expression, ...
... [ FROM table-expression, ]
... [ WHERE search-condition ]
... [ ORDER BY expression [ ASC | DESC ] , ...]
FROM table-expression
```

table-expression:

```
table-spec
| table-expression join-type table-spec [ ON condition ]
| table-expression, ...
```

Examples

- **Example 1** – transfer employee Philip Chin (employee 129) from the sales department to the marketing department:

```
UPDATE Employees
SET DepartmentID = 400
WHERE EmployeeID = 129;
```

- **Example 2** – the Marketing Department (400) increases bonuses from 4% to 6% of each employee's base salary:

```
UPDATE Employees
SET bonus = base * 6/100
WHERE DepartmentID =400;
```

- **Example 3** – each employee gets a pay increase with the department bonus:

```
UPDATE Employees
SET emp.Salary = emp.Salary + dept.bonus
FROM Employees emp, Departments dept
WHERE emp.DepartmentID = dept.DepartmentID;
```

- **Example 4** – another way to give each employee a pay increase with the department bonus:

```
UPDATE Employees
SET emp.salary = emp.salary + dept.bonus
FROM Employees emp JOIN Departments dept
ON emp.DepartmentID = dept.DepartmentID;
```

Usage

The table on which you use **UPDATE** may be a base table or a temporary table.

Defaults on updates are honored for current user, user and current timestamp, and timestamp only.

Each named column is set to the value of the expression on the right-hand side of the equal sign. Even *column-name* can be used in the expression—the old value is used.

The **FROM** clause can contain multiple tables with join conditions and returns all the columns from all the tables specified and filtered by the join condition and/or **WHERE** condition.

Using the wrong join condition in a **FROM** clause causes unpredictable results. If the **FROM** clause specifies a one-to-many join and the **SET** clause references a cell from the “many” side of the join, the cell is updated from the first value selected. In other words, if the join condition causes multiple rows of the table to be updated per row ID, the first row returned becomes the update result. For example:

```
UPDATE T1
SET T1.c2 = T2.c2
FROM T1 JOIN TO T2
ON T1.c1 = T2.c1
```

If table T2 has more than one row per T2 . c1, results might be as follows:

T2 . c1	T2 . c2	T2 . c3
1	4	3
1	8	1
1	6	4
1	5	2

With no **ORDER BY** clause, T1 . c2 may be 4, 6, 8, or 9.

- With **ORDER BY T2 . c3**, T1 . c2 is updated to 8.
- With **ORDER BY T2 . c3 DESC**, T1 . c2 is updated to 6.

SAP Sybase IQ rejects any **UPDATE** statement in which the table being updated is on the null-supplying side of an outer join. In other words:

- In a left outer join, the table on the left side of the join cannot be missing any rows on joined columns.
- In a right outer join, the table on the right side of the join cannot be missing any rows on joined columns.
- In a full outer join, neither table can be missing any rows on joined columns.

For example, in this statement, table T1 is on the left side of a left outer join, and thus cannot contain be missing any rows:

```
UPDATE T1
SET T1.c2 = T2.c4
```

```
FROM T1 LEFT OUTER JOIN T2
ON T1.rowid = T2.rowid
```

Normally, the order in which rows are updated does not matter. However, in conjunction with the **NUMBER(*)** function, an ordering can be useful to get increasing numbers added to the rows in some specified order. If you are not using the **NUMBER(*)** function, avoid using the **ORDER BY** clause, because the **UPDATE** statement performs better without it.

In an **UPDATE** statement, if the **NUMBER(*)** function is used in the **SET** clause and the **FROM** clause specifies a one-to-many join, **NUMBER(*)** generates unique numbers that increase, but do not increment sequentially due to row elimination.

You can use the **ORDER BY** clause to control the result from an **UPDATE** when the **FROM** clause contains multiple joined tables.

SAP Sybase IQ ignores the **ORDER BY** clause in searched **UPDATE** and returns a message that the syntax is not valid ANSI syntax.

If no **WHERE** clause is specified, every row is updated. If you specify a **WHERE** clause, SAP Sybase IQ updates only rows satisfying the search condition.

The left side of each **SET** clause must be a column in a base table.

Views can be updated provided the **SELECT** statement defining the view does not contain a **GROUP BY** clause or an aggregate function, or involve a **UNION** operation. The view should contain only one table.

Character strings inserted into tables are always stored in the case they are entered, regardless of whether the database is case-sensitive or not. Thus a character data type column updated with the string 'Value' is always held in the database with an uppercase V and the remainder of the letters lowercase. **SELECT** statements return the string as 'Value.' If the database is not case-sensitive, however, all comparisons make 'Value' the same as 'value,' 'VALUE,' and so on. The IQ server may return results in any combination of lowercase and uppercase, so you cannot expect case-sensitive results in a database that is case-insensitive (**CASE IGNORE**). Further, if a single-column primary key already contains an entry 'Value,' an **INSERT** of 'value' is rejected, as it would make the primary key not unique.

If the update violates any check constraints, the whole statement is rolled back.

SAP Sybase IQ supports scalar subqueries within the **SET** clause, for example:

```
UPDATE r
SET r.o= (SELECT MAX(t.o)
FROM t ... WHERE t.y = r.y),
r.s= (SELECT SUM(x.s)
FROM x ...
WHERE x.x = r.x)
WHERE r.a = 10
```

SAP Sybase IQ supports **DEFAULT** column values in **UPDATE** statements. If a column has a **DEFAULT** value, this **DEFAULT** value is used as the value of the column in any **UPDATE** statement that does not explicitly modify the value for the column.

See *CREATE TABLE Statement* for details about updating IDENTITY/AUTOINCREMENT columns, which are another type of DEFAULT column.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—With these exceptions, syntax of the IQ **UPDATE** statement is generally compatible with the Adaptive Server Enterprise **UPDATE** statement Syntax 1: SAP Sybase IQ supports multiple tables with join conditions in the **FROM** clause. Updates of remote tables are limited to SAP Sybase IQ syntax supported by CIS.

Permissions

Requires UPDATE privilege on the columns being modified.

See also

- *CREATE TABLE Statement* on page 205

UPDATE (positioned) Statement [ESQL] [SP]

Modifies the data at the current location of a cursor.

Syntax

```
UPDATE table-list
  SET set-item, ...
  WHERE CURRENT OF cursor-name

set-item:
  column-name [.field-name...] = scalar-value
```

Parameters

- **cursor-name** – identifier or hostvar
- **SET** – the columns that are referenced in *set-item* must be in the base table that is updated. They cannot refer to aliases, nor to columns from other tables or views. If the table you are updating is given a correlation name in the cursor specification, you must use the correlation name in the SET clause. The expression on the right side of the SET clause may reference columns, constants, variables, and expressions from the **SELECT** clause of the query.
- **set-item** – expression cannot contain functions or expressions.

- **WHERE CURRENT OF** – use of ORDER BY in the WHERE CURRENT OF clause is not recommended. The ORDER BY columns may be updated, but the result set does not reorder. The results appear to fetch out of order and appear to be incorrect.

Examples

- **Example 1** – an **UPDATE** statement using WHERE CURRENT OF cursor:

```
UPDATE Employees SET surname = 'Jones'  
WHERE CURRENT OF emp_cursor
```

Usage

This form of the **UPDATE** statement updates the current row of the specified cursor. The current row is defined to be the last row successfully fetched from the cursor, and the last operation on the cursor cannot have been a positioned **DELETE** statement.

The requested columns are set to the specified values for the row at the current row of the specified query. The columns must be in the select list of the specified open cursor.

Changes effected by positioned **UPDATE** statements are visible in the cursor result set, except where client-side caching prevents seeing these changes. Rows that are updated so that they no longer meet the requirements of the WHERE clause of the open cursor are still visible.

Since SAP Sybase IQ does not support the **CREATE VIEW... WITH CHECK OPTION**, positioned **UPDATE** does not support this option. The WITH CHECK OPTION clause does not allow an update that creates a row that is not visible by the view.

A rowid column cannot be updated by a positioned **UPDATE**.

SAP Sybase IQ supports repeatedly updating the same row in the result set.

Standards

- The range of cursors that can be updated may contain vendor extensions to ISO/ANSI SQL grammar if the ANSI_UPDATE_CONSTRAINTS option is set to OFF.
- Embedded SQL use is supported by Open Client/Open Server, and procedure and trigger use is supported in SQL Anywhere.

Permissions

Must have UPDATE permission on the columns being modified.

See also

- *DECLARE CURSOR Statement [ESQL] [SP]* on page 240
- *DELETE Statement* on page 248
- *DELETE (positioned) Statement [ESQL] [SP]* on page 250

- *UPDATE Statement* on page 448

VALIDATE Statement

Validates the current database, or a single table, materialized view, or index in the IQ catalog (system) store.

Warning! Validating a table or an entire database should be performed while no connections are making changes to the database; otherwise, errors may be reported indicating some form of database corruption even though no corruption actually exists.

Syntax

Syntax 1 – Validating a database

```
VALIDATE { CHECKSUM | DATABASE }
```

Syntax 2 – Validating tables and materialized views

```
VALIDATE {
  TABLE [ owner.]table-name
  | MATERIALIZED VIEW [ owner.]materialized-view-name }
[ WITH EXPRESS CHECK ]
```

Syntax 3 – Validating indexes

```
VALIDATE {
  INDEX index-name
  | [ INDEX ] FOREIGN KEY role-name
  | [ INDEX ] PRIMARY KEY }
ON [ owner.]object-name
```

object-name:
table-name | materialized-view-name

Syntax 4 – Validating text indexes

```
VALIDATE TEXT INDEX index-name
ON [ owner.]table-name
```

Parameters

- **CHECKSUM** – validates the checksum on each page of a database. The CHECKSUM clause ensures that database pages have not been modified on disk. When a database is created with checksums enabled, a checksum is calculated for each database page before it is written to disk. CHECKSUM reads each database page directly from disk—not via the database server's cache—and calculates the checksum for each page. If the calculated checksum for a page does not match the stored checksum for that page, an error occurs and information about the invalid page appears in the database server messages window.

The CHECKSUM clause is not recommended for databases that have checksums disabled because it reads the entire database from disk.

- **DATABASE** – ensures that the free map correctly identifies pages as either allocated or free and that no BLOBs have been orphaned. The DATABASE clause also performs checksum validation and verifies that each database page belongs to the correct object. For example, on a table page, the table ID must identify a valid table whose definition must include the current page in its set of table pages.

The DATABASE clause brings pages into the database server's cache in sequential order. This results in their validation, as the database server always verifies the contents and checksums of pages brought into the cache. If you start database validation while the database cleaner is running, the validation does not run until the database cleaner is finished running.

- **TABLE** – validates the specified table and all of its indexes by checking that the set of all rows and values in the base table matches the set of rows and values contained in each index. The TABLE clause also traverses all the table's BLOBs, verifies BLOB allocation maps, and detects orphaned BLOBs. The TABLE clause checks the physical structure of the table's index pages and verifies the order of the index hash values, and the index's uniqueness requirements (if any are specified).

For foreign key indexes, unless the WITH EXPRESS CHECK clause is specified, each value is looked up in the primary key table to verify that referential integrity is intact. Because the TABLE clause, like the DATABASE clause, uses the database server's cache, the database server also verifies the checksums and basic validity of all pages in use by a table and its indexes.

- **INDEX** – performs the same operations as the TABLE clause except that it only validates the specified index and its underlying table; other indexes are not checked.

For foreign key indexes, unless the WITH EXPRESS CHECK clause is specified, each value is looked up in the primary key table to verify that referential integrity is intact. Specifying the WITH EXPRESS CHECK clause disables referential integrity checking and can therefore significantly improve performance. If the specified index is not a foreign key index, WITH EXPRESS CHECK has no effect.

- **TEXT INDEX** – verifies that the positional information for the terms in the index is intact. If the positional information is not intact, an error is generated and you must rebuild the text index. If the text index is either auto or manual, you can rebuild the text index by executing the **REFRESH TEXT INDEX** statement. If the generated error concerns an immediate text index, you must drop the immediate index and create a new one.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires one of:

- VALIDATE ANY OBJECT system privilege.

VALIDATE LDAP SERVER Statement

Validates changes to the settings of existing LDAP server configuration objects before applying them.

Syntax

```
VALIDATE LDAP SERVER [ ldapua-server-name | ldapua-server-attrs ]
  [ CHECK userid [ user-dn-string ] ]
```

ldapua-server-attrs:

SEARCH DN

```
    URL { 'URL_string' | NULL }
    | ACCESS ACCOUNT { 'DN_string' | NULL }
    | IDENTIFIED BY ( 'password' | NULL }
    | IDENTIFIED BY ENCRYPTED { encrypted-password | NULL }
```

```
    | AUTHENTICATION URL { 'URL_string' | NULL }
    | CONNECTION TIMEOUT timeout_value
    | CONNECTION RETRIES retry_value
    | TLS { ON | OFF }
```

Parameters

- **ldapua-server-name** – identifies the LDAP server configuration object.
- **URL** – identifies the host (by name or by IP address), port number, and the search to be performed for the DN lookup for a given user ID. This value is validated for correct LDAP URL syntax before it is stored in the ISYSLDAPSERVER system table. The maximum size for this string is 1024 bytes.
- **ACCESS ACCOUNT** – a user created on the LDAP server for use by SAP Sybase IQ, not a user within SAP Sybase IQ. The distinguished name (DN) for this user is used to connect to the LDAP server. This user has permissions within the LDAP server to search for DNs by user ID in the locations specified by the SEARCH DN URL. The maximum size for this string is 1024 bytes.
- **IDENTIFIED BY** – provides the password associated with the ACCESS ACCOUNT user. The password is stored using symmetric encryption on disk. Use the value NULL to clear the password and set it to none. The maximum size of a clear text password is 255 bytes.
- **IDENTIFIED BY ENCRYPTED** – configures the password associated with the ACCESS ACCOUNT distinguished name in an encrypted format. The binary value is the

encrypted password and is stored on disk as is. Use the value NULL to clear the password and set it to none. The maximum size of the binary is 289 bytes

- **AUTHENTICATION URL** – identifies the host (by name or IP address) and the port number of the LDAP server to use for authentication of the user. This is the value defined for <URL_string> and is validated for correct LDAP URL syntax before it is stored in ISYSLDAPSERVER system table. The DN of the user obtained from a prior DN search and the user password bind a new connection to the authentication URL. A successful connection to the LDAP server is considered proof of the identity of the connecting user. The maximum size for this string is 1024 bytes.
- **CONNECTION TIMEOUT** – specifies the connection timeout from SAP Sybase IQ to the LDAP server for both DN searches and authentication. This value is in milliseconds, with a default value of 10 seconds.
- **CONNECTION RETRIES** – specifies the number of retries on connections from SAP Sybase IQ to the LDAP server for both DN searches and authentication. The valid range of values is 1 – 60, with a default value of 3.
- **TLS** – defines whether the TLS or Secure LDAP protocol is used for connections to the LDAP server for both DN searches and authentication. When set to ON, the TLS protocol is used and the URL begins with "ldap://" When set to OFF (or not specified), Secure LDAP protocol is used and the URL begins with "ldaps://". When using the TLS protocol, specify the database security option TRUSTED_CERTIFICATES_FILE with a file name containing the certificate of the Certificate Authority (CA) that signed the certificate used by the LDAP server.
- **CHECK userID** – the userID whose existence is validated on the LDAP server.
- **user-dn-string** – compares a user's DN value with the user ID for verification purposes.

Examples

- **Example 1** – assume the apps_primary LDAP server configuration object was created as follows:

```
SET OPTION PUBLIC.login_mode = 'Standard,LDAPUA'  
CREATE LDAP SERVER apps_primary  
SEARCH DN  
    URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'  
    ACCESS ACCOUNT 'cn=aseadmin, cn=Users, dc=mycompany, dc=com'  
    IDENTIFIED BY 'Secret99Password'  
AUTHENTICATION URL 'ldap://my_LDAPserver:389/'  
CONNECTION TIMEOUT 3000  
WITH ACTIVATE
```

This statement validates the existence of a userID myusername by using the optional CHECK clause to compare the userID to the expected user distinguished name (enclosed in quotation marks) on the apps_primary LDAP server configuration object.

```
VALIDATE LDAP SERVER apps_primary
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com'
```

- **Example 2** – the name of the LDAP server configuration object does not have to be defined in the **VALIDATE LDAP SERVER** statement if you include the search attributes:

```
VALIDATE LDAP SERVER
SEARCH DN
  URL 'ldap://my_LDAPserver:389/dc=MyCompany,dc=com??sub?cn=*'
  ACCESS ACCOUNT 'cn=aseadmin,cn=Users,dc=mycompany,dc=com'
  IDENTIFIED BY 'Secret99Password'
AUTHENTICATION URL 'ldap://my_LDAPserver:389/'
CONNECTION TIMEOUT 3000
CHECK myusername 'cn=myusername,cn=Users,dc=mycompany,dc=com'
```

Usage

This statement is useful for an administrator when setting up a new server to use LDAP user authentication, and for diagnosing problems between the LDAP server configuration object and the external LDAP server. Any connection made by the **VALIDATE LDAP SERVER** statement is temporary and is closed by the end of the statement.

When validating the LDAP server configuration object by name, definitions from prior **CREATE LDAP SERVER** and **ALTER LDAP SERVER** statements are used. Alternately, when *ldapua-server-attributes* are specified instead of the LDAP server configuration object name, the specified attributes are validated. When *ldapua-server-attributes* are specified, the URLs are parsed to identify syntax errors, and statement processing stops if a syntax error is detected,

Whether using an LDAP server configuration object name or a successfully parsed set of *ldapua-server-attributes*, a connection to the external LDAP server is attempted. If the parameter **ACCESS ACCOUNT** and a password are specified, the values are used to establish the connection to the **SEARCH DN** URL. This the **SEARCH DN** URL, **ACCESS ACCOUNT**, and **ACCESS ACCOUNT** password.

When using the optional **CHECK** clause, the **userID** is used in the search to validate the existence of the user on the external LDAP server. When the expected DN value for a given user is known, the value can be specified, and is compared with the result of the search to determine success or failure.

Standards

ANSI SQL – Compliance level: Transact-SQL extension.

Permissions

Requires the **MANAGE ANY LDAP SERVER** system privilege.

WAITFOR Statement

Delays processing for the current connection for a specified amount of time or until a given time.

Syntax

```
WAITFOR {  
    DELAY time_value | TIME time_value }  
    [ CHECK EVERY integer ]  
    [ AFTER MESSAGE BREAK ]
```

Parameters

- **DELAY** – processing is suspended for the given *time_value* interval.
- **TIME** – processing is suspended until the server time reaches the *time_value* specified.
- **time_value** – string
- **CHECK EVERY** – controls how often the **WAITFOR** statement wakes up. By default, **WAITFOR** wakes up every 5 seconds. The value is in milliseconds, and the minimum value is 250milliseconds.
- **AFTER MESSAGE BREAK** – the **WAITFOR** statement can be used to wait for a message from another connection. In most cases, when a message is received it is forwarded to the application that executed the **WAITFOR** statement and the **WAITFOR** statement continues to wait. If the **AFTER MESSAGE BREAK** clause is specified, when a message is received from another connection, the **WAITFOR** statement completes. The message text is not forwarded to the application, but it can be accessed by obtaining the value of the `MessageReceived` connection property.

Examples

- **Example 1** – wait for three seconds:

```
WAITFOR DELAY '00:00:03'
```

- **Example 2** – wait for 0.5 seconds (500 milliseconds):

```
WAITFOR DELAY '00:00:00:500'
```

- **Example 3** – wait until 8 p.m.:

```
WAITFOR TIME '20:00'
```

Usage

The **WAITFOR** statement wakes up periodically (every 5 seconds by default) to check if it has been canceled or if messages have been received. If neither of these has happened, the statement continues to wait.

If the current server time is greater than the time specified, processing is suspended until that time on the following day.

WAITFOR provides an alternative to the following statement, and might be useful for customers who choose not to enable Java in the database:

```
call java.lang.Thread.sleep( <time_to_wait_in_millisecs> )
```

In many cases, scheduled events are a better choice than using **WAITFOR TIME**, because scheduled events execute on their own connection.

Side Effects

- The implementation of this statement uses a worker thread while it is waiting. This uses up one of the threads specified by the `-gn` server command line option.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—This statement is also implemented by Adaptive Server Enterprise.

Permissions

None

See also

- *CREATE EVENT Statement* on page 123

WHENEVER Statement [ESQL]

Specifies error handling in an Embedded SQL program.

Syntax

```
WHENEVER
  { SQLERROR | SQLWARNING | NOTFOUND }
  ... { GOTO label | STOP | CONTINUE | C code; }
```

Examples

- **Example 1** – This example executes done when the NOTFOUND clause is met:

```
EXEC SQL WHENEVER NOTFOUND GOTO done;
```

SQL Statements

- **Example 2** – this example uses the `SQLERROR` clause:

```
EXEC SQL WHENEVER SQLERROR
{
    PrintError( &sqlca );
    return( FALSE );
};
```

Usage

WHENEVER can be put anywhere in an Embedded SQL C program, and does not generate any code. The preprocessor generates code following each successive SQL statement. The error action remains in effect for all Embedded SQL statements from the source line of the **WHENEVER** statement until the next **WHENEVER** statement with the same error condition, or the end of the source file.

The default action is **CONTINUE**.

WHENEVER is provided for convenience in simple programs. Most of the time, checking the `sqlcode` field of the `SQLCA` (`SQLCODE`) directly is the easiest way to check error conditions. In this case, **WHENEVER** is not used. The **WHENEVER** statement causes the preprocessor to generate an *if*(`SQLCODE`) test after each statement.

Note: The error conditions are in effect based on positioning in the C language source file and not on when the statements are executed.

Standards

- SQL—Vendor extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Open Client/Open Server.

Permissions

None

WHILE Statement [T-SQL]

Provides repeated execution of a statement or compound statement.

Syntax

```
WHILE expression
... statement
```

Examples

- **Example 1** – the **BREAK** statement breaks the **WHILE** loop, if the most expensive product has a price less than \$50. Otherwise, the loop continues until the average price is greater than \$30.

```
WHILE (SELECT AVG(unit_price) FROM Products) < 30
BEGIN
    DELETE FROM Products
    WHERE UnitPrice = MAX(UnitPrice)
    IF ( SELECT MAX(UnitPrice) FROM Products ) < 50
        BREAK
END
```

Usage

The **WHILE** conditional affects the performance of only a single SQL statement, unless statements are grouped into a compound statement between the keywords **BEGIN** and **END**.

The **BREAK** statement and **CONTINUE** statement can be used to control execution of the statements in the compound statement. The **BREAK** statement terminates the loop, and execution resumes after the **END** keyword, marking the end of the loop. The **CONTINUE** statement causes the **WHILE** loop to restart, skipping any statements after the **CONTINUE**.

Standards

- SQL—Transact-SQL extension to ISO/ANSI SQL grammar.
- Sybase—Supported by Adaptive Server Enterprise.

Permissions

None

See also

- *BEGIN ... END Statement* on page 84

Database Options

Database options and Interactive SQL options customize and modify database behavior. SAP Sybase IQ database options are divided into three classes: general, Transact-SQL compatibility, and Interactive SQL.

Introduction to Database Options

Database options control many aspects of database behavior including compatibility, error handling, and concurrency.

For example, you can use database options for the purposes such as:

- Compatibility – lets you control how much like Adaptive Server Enterprise your SAP Sybase IQ database operates, and whether SQL that does not conform to SQL92 generates errors.
- Error handling – lets you control what happens when errors, such as dividing by zero or overflow errors, occur.
- Concurrency and transactions – lets you control the degree of concurrency and details of COMMIT behavior using options.

You set options with the **SET OPTION** statement, which has this general syntax:

```
SET [ EXISTING ] [ TEMPORARY ] OPTION
... [ userid. | PUBLIC. ] option-name = [ option-value ]
```

Specify a user ID or role name to set the option only for that user or role. Every user belongs to the **PUBLIC** role. If no user ID or role is specified, the option change is applied to the currently logged on user ID that issued the **SET OPTION** statement.

For example, this statement applies a change to the **PUBLIC** user ID, a role to which all users belong:

```
SET OPTION Public.login_mode = standard
```

Note: When you set an option to **TEMPORARY** without specifying a user or role, the new option value takes effect only for the currently logged on user ID that issued the statement, and only for the duration of the connection. When you set an option to **TEMPORARY** for the **PUBLIC** role, the change remains in place for as long as the database is running—when the database shuts down, **TEMPORARY** options for the **PUBLIC** role revert back to their permanent value.

When you set an option without issuing the **TEMPORARY** keyword, the new option value is permanent for the user or role who issued the statement.

See *Scope and Duration of Database Options*, *Temporary Options*, and *SET OPTION Statement* for more information on temporary versus permanent option values.

The maximum length of *option-value*, when set to a string, is 127 bytes.

Note: For all database options that accept integer values, SAP Sybase IQ truncates any decimal *option-value* setting to an integer value. For example, the value 3.8 is truncated to 3.

Warning! Do not change option settings while fetching rows.

See also

- *Scope and Duration of Database Options* on page 465
- *Temporary Options* on page 466
- *SET OPTION Statement* on page 431

Current Option Settings

You can obtain a list of option settings, or the values of individual options, using **sp_iqcheckoptions**, **sa_conn_properties**, the **SET** statement, Sybase Control Center, and the SYSOPTIONS system view.

- For the connected user, the **sp_iqcheckoptions** stored procedure displays a list of the current value and the default value of database options that have been changed from the default. **sp_iqcheckoptions** considers all SAP Sybase IQ and SQL Anywhere database options. SAP Sybase IQ modifies some SQL Anywhere option defaults, and these modified values become the new default values. Unless the new SAP Sybase IQ default value is changed again, **sp_iqcheckoptions** does not list the option.

sp_iqcheckoptions also lists server start-up options that have been changed from the default values.

When a DBA runs **sp_iqcheckoptions**, he or she sees all options set on a permanent basis for all roles and users and sees temporary options set for DBA. Users who are not DBAs see their own temporary options. All users see nondefault server start-up options.

The **sp_iqcheckoptions** stored procedure requires no parameters. In Interactive SQL, run:

```
sp_iqcheckoptions
```

The system table DBA.SYSOPTIONDEFAULTS contains all of the names and default values of the SAP Sybase IQ and SQL Anywhere options. You can query this table to see all option default values.

- Current option settings for your connection are available as a subset of connection properties. You can list all connection properties using the **sa_conn_properties** system procedure:

```
call sa_conn_properties
```

- In Interactive SQL, the **SET** statement with no arguments lists the current setting of options:

```
SET
```

- In Sybase Control Center, right-click a database and select Options from the submenu.
- Query the SYSOPTIONS system view:

```
SELECT *
FROM SYSOPTIONS
```

This shows all PUBLIC values, and those USER values that have been explicitly set.

- Use the **connection_property** system function to obtain an individual option setting. For example, this statement returns the value of the Ansinull option:

```
SELECT connection_property ('Ansinull')
```

Scope and Duration of Database Options

You can set options at three levels of scope: public, user, and temporary.

Temporary options take precedence over user and public settings. User-level options take precedence over public settings. If you set a user-level option for the current user, the corresponding temporary option is set as well.

Some options, such as COMMIT behavior, are database-wide in scope. Setting these options requires DBA permissions. Other options, such as ISOLATION_LEVEL, can also be applied to only the current connection, and need no special permissions.

Changes to option settings take place at different times, depending on the option. Changing a global option such as RECOVERY_TIME takes place the next time the server is started. Some of the options that take effect after the server is restarted:

Database Options that Require Restarting the Server
CACHE_PARTITIONS
CHECKPOINT_TIME
OS_FILE_CACHE_BUFFERING
OS_FILE_CACHE_BUFFERING_TEMPDB
PREFETCH_BUFFER_LIMIT
PREFETCH_BUFFER_PERCENT
RECOVERY_TIME
SWEEPER_THREADS_PERCENT
WASH_AREA_BUFFERS_PERCENT

Options that affect only the current connection generally take place immediately. For example, you can change option settings in the middle of a transaction.

Warning! Changing options when a cursor is open can lead to unreliable results. For example, changing DATE_FORMAT might not change the format for the next row when a cursor is

opened. Depending on the way the cursor is being retrieved, it might take several rows before the change works its way to the user.

Temporary Options

Adding the **TEMPORARY** keyword to the `SET OPTION` statement changes the duration of the change.

Ordinarily an option change is permanent: it will not change until it is explicitly changed using the **SET OPTION** statement.

When the **SET TEMPORARY OPTION** statement is executed, the new option value takes effect only for the current connection, and only for the duration of the connection.

When the **SET TEMPORARY OPTION** is used to set a **PUBLIC** option, the change is in place for as long as the database is running. When the database is shut down, **TEMPORARY** options for the **PUBLIC** user ID revert back to their permanent value.

Setting an option for the **PUBLIC** user ID temporarily offers a security advantage. For example, when the `LOGIN_MODE` option is enabled, the database relies on the login security of the system on which it is running. Enabling `LOGIN_MODE` temporarily means that a database relying on the security of a Windows domain will not be compromised if the database is shut down and copied to a local machine. In this case, the `LOGIN_MODE` option reverts to its permanent value, which could be `Standard`, a mode where integrated logins are not permitted.

PUBLIC Options

PUBLIC options can be set for a user, user extended role, or the **PUBLIC** role. They can be set for self or for another user or role.

Setting a **PUBLIC** option for the **PUBLIC** role sets the value for all users who do not already have the **PUBLIC** option set at the user level. Setting a **PUBLIC** option for a user or user-extended role overrides any value defined at the **PUBLIC** role level.

No system privilege is required to set a **PUBLIC** option for self, but does require the `SET ANY PUBLIC OPTION` system privilege to set for another user, user-extended role, or **PUBLIC** role. **PUBLIC** options cannot be set for user-defined roles. **PUBLIC** database options take effect immediately. No shut down and restart of the database server is required for the change to take effect.

SECURITY Options

SECURITY options are special category which are relevant to security of the database. It can be set at user level or **PUBLIC** level depending on options.

Changes to **SECURITY** database options take effect immediately. Requires the `SET ANY SECURITY OPTION` system privilege to set **SECURITY** database options.

No shut down and restart of the database server is required for the change to take effect.

SYSTEM Options

SYSTEM options are special category which are relevant to security of the database. It can be set at user level or PUBLIC level.

Requires the SET ANY SYSTEM OPTION system privilege to set SYSTEM options. Takes effect immediately.

Delete an Option Setting

Omit the *option-value* to delete the option setting from the database.

If *option-value* is omitted, the specified option setting is deleted from the database. If *option-value* is a personal option setting, the value reverts back to the PUBLIC setting. If a TEMPORARY option is deleted, the option setting reverts back to the permanent setting.

For example, reset the ANSINULL option to its default value:

```
SET OPTION ANSINULL =
```

If you incorrectly type the name of an option when you are setting the option, the incorrect name is saved in the SYSOPTION table. You can remove the incorrectly typed name from the SYSOPTION table by setting the option PUBLIC with an equality after the option name and no value:

```
SET OPTION PUBLIC.a_mistyped_name=;
```

For example, if you set an option and incorrectly type the name, you can verify that the option was saved by selecting from the SYSOPTIONS view:

```
SET OPTION PUBLIC.a_mistyped_name='ON';
SELECT * FROM SYSOPTIONS ORDER BY 2;
```

user_name	option	setting
PUBLIC	a_mistyped_name	ON
PUBLIC	Abort_On_Error_File	
PUBLIC	Abort_On_Error_Line	0
PUBLIC	Abort_On_Error_Number	0
...		

Remove the incorrectly typed option by setting the option to no value, then verify that the option is removed:

```
SET OPTION PUBLIC.a_mistyped_name=;
SELECT * FROM SYSOPTIONS ORDER BY 2;
```

user_name	option	setting
PUBLIC	Abort_On_Error_File	
PUBLIC	Abort_On_Error_Line	0
PUBLIC	Abort_On_Error_Number	0
...		

If you remove the PUBLIC option and then try to add the USER option, an error message displays:

```

Couldn't execute the statement.
Invalid option 'chained' -- no PUBLIC setting exists
SQLCODE=-200?ODBC 3 State="42000"
Line 1,Column 29
    
```

In order to reset the PUBLIC option to the default value, explicitly set the default value:

```

SET OPTION PUBLIC.chained = 'ON';
    
```

Initial Option Settings

You can use stored procedures to configure the initial database option settings of a user.

Connections to SAP Sybase IQ can be made through the TDS (tabular data stream) protocol (Open Client and jConnect™ for JDBC™ connections) or through the SAP Sybase IQ protocol (ODBC, Embedded SQL).

If users have both TDS and the SAP Sybase IQ-specific protocol, you can configure their initial settings using stored procedures. As it is shipped, SAP Sybase IQ uses this method to set Open Client connections and jConnect connections to reflect default Adaptive Server Enterprise behavior.

The initial settings are controlled using the LOGIN_PROCEDURE option, which is called after all the checks have been performed to verify that the connection is valid. The LOGIN_PROCEDURE option names a stored procedure to run when users connect. The default setting is to use the sp_login_environment system stored procedure. You can specify a different stored procedure.

The sp_login_environment procedure checks to see if the connection is being made over TDS. If it is, it calls the sp_tsql_environment procedure, which sets several options to new default values for the current connection.

See also

- *LOGIN_PROCEDURE Option* on page 572

Deprecated Database Options

See New Features Summary SAP Sybase IQ 16.0 for information about database options deprecated in this release.

General Database Options

General database options is the class of options consisting of all options except Transact-SQL compatibility options and Interactive SQL options.

See also

- *Transact-SQL Compatibility Options* on page 474
- *Interactive SQL Options* on page 476
- *AFFINITY_AUTOEXCLUDE_TIMEOUT Option* on page 477
- *AGGREGATION_PREFERENCE Option* on page 478
- *ALLOW_SNAPSHOT_VERSIONING Option* on page 480
- *ANSI_UPDATE_CONSTRAINTS Option* on page 484
- *ALLOW_READ_CLIENT_FILE Option* on page 485
- *ASE_BINARY_DISPLAY Option* on page 486
- *ASE_FUNCTION_BEHAVIOR Option* on page 487
- *AUDITING Option [database]* on page 488
- *BASE_TABLES_IN_RLV_STORE Option* on page 488
- *BIT_VECTOR_PINNABLE_CACHE_PERCENT Option* on page 489
- *BLOCKING Option* on page 490
- *BLOCKING_TIMEOUT Option* on page 490
- *BT_PREFETCH_MAX_MISS Option* on page 491
- *BT_PREFETCH_SIZE Option* on page 492
- *BTREE_PAGE_SPLIT_PAD_PERCENT Option* on page 493
- *CACHE_AFFINITY_PERCENT Option* on page 494
- *CACHE_PARTITIONS Option* on page 494
- *CHECKPOINT_TIME Option* on page 496
- *CIS_ROWSET_SIZE Option* on page 497
- *CONVERSION_MODE Option* on page 499
- *CONVERT_VARCHAR_TO_1242 Option* on page 506
- *COOPERATIVE_COMMIT_TIMEOUT Option* on page 506
- *COOPERATIVE_COMMITS Option* on page 507
- *CREATE_HG_WITH_EXACT_DISTINCTS* on page 508
- *CURSOR_WINDOW_ROWS Option* on page 510
- *DATE_FIRST_DAY_OF_WEEK Option* on page 510

- *DATE_FORMAT Option* on page 512
- *DATE_ORDER Option* on page 514
- *DBCC_LOG_PROGRESS Option* on page 514
- *DBCC_PINNABLE_CACHE_PERCENT Option* on page 515
- *DEBUG_MESSAGES Option* on page 516
- *DEDICATED_TASK Option* on page 517
- *DEFAULT_DBSPACE Option* on page 517
- *DEFAULT_DISK_STRIPING Option* on page 519
- *DEFAULT_HAVING_SELECTIVITY_PPM Option* on page 519
- *DEFAULT_KB_PER_STRIPE Option* on page 521
- *DEFAULT_LIKE_MATCH_SELECTIVITY_PPM Option* on page 522
- *DEFAULT_LIKE_RANGE_SELECTIVITY_PPM Option* on page 523
- *DEFAULT_PROXY_TABLE_ROW_COUNT Option* on page 523
- *DEFAULT_TABLE_UDF_ROW_COUNT Option* on page 524
- *DELAYED_COMMIT_TIMEOUT Option* on page 525
- *DELAYED_COMMITS Option* on page 525
- *DISABLE_RI_CHECK Option* on page 526
- *DQP_ENABLED Option* on page 527
- *DQP_ENABLED_OVER_NETWORK Option* on page 528
- *EARLY_PREDICATE_EXECUTION Option* on page 528
- *ENABLE_ASYNC_IO Option* on page 530
- *ENABLE_LOB_VARIABLES Option* on page 530
- *EXTENDED_JOIN_SYNTAX Option* on page 531
- *FLOATING_POINT_ACCUMULATOR Option* on page 532
- *FORCE_DROP Option* on page 532
- *FORCE_NO_SCROLL_CURSORS Option* on page 533
- *FORCE_UPDATABLE_CURSORS Option* on page 534
- *FP_LOOKUP_SIZE Option* on page 535
- *FP_LOOKUP_SIZE_PPM Option* on page 535
- *FP_NBIT_AUTOSIZE_LIMIT Option* on page 536
- *FP_NBIT_ENFORCE_LIMITS Option* on page 538
- *FP_NBIT_IQ15_COMPATIBILITY Option* on page 539
- *FP_NBIT_LOOKUP_MB Option* on page 540
- *FP_NBIT_ROLLOVER_MAX_MB Option* on page 542
- *FP_PREDICATE_WORKUNIT_PAGES Option* on page 543
- *FPL_EXPRESSION_MEMORY_KB Option* on page 543
- *GARRAY_FILL_FACTOR_PERCENT Option* on page 544
- *GARRAY_INSERT_PREFETCH_SIZE Option* on page 545
- *GARRAY_PAGE_SPLIT_PAD_PERCENT Option* on page 545

- *GARRAY_RO_PREFETCH_SIZE* Option on page 546
- *HASH_PINNABLE_CACHE_PERCENT* Option on page 547
- *HASH_THRASHING_PERCENT* Option on page 548
- *HG_DELETE_METHOD* Option on page 548
- *HG_SEARCH_RANGE* Option on page 549
- *HTTP_SESSION_TIMEOUT* Option on page 550
- *IDENTITY_ENFORCE_UNIQUENESS* Option on page 551
- *IDENTITY_INSERT* Option on page 551
- *IN_SUBQUERY_PREFERENCE* Option on page 552
- *INDEX_ADVISOR* Option on page 554
- *INDEX_ADVISOR_MAX_ROWS* Option on page 556
- *INDEX_PREFERENCE* Option on page 557
- *INFER_SUBQUERY_PREDICATES* Option on page 558
- *IQGOVERN_MAX_PRIORITY* Option on page 559
- *IQGOVERN_PRIORITY* Option on page 559
- *IQGOVERN_PRIORITY_TIME* Option on page 560
- *ISOLATION_LEVEL* Option on page 561
- *JAVA_LOCATION* Option on page 561
- *JAVA_VM_OPTIONS* Option on page 562
- *JOIN_EXPANSION_FACTOR* Option on page 562
- *JOIN_OPTIMIZATION* Option on page 563
- *JOIN_PREFERENCE* Option on page 565
- *JOIN_SIMPLIFICATION_THRESHOLD* Option on page 567
- *LF_BITMAP_CACHE_KB* Option on page 567
- *LOAD_ZEROLENGTH_ASNULL* Option on page 568
- *LOG_CONNECT* Option on page 569
- *LOG_CURSOR_OPERATIONS* Option on page 570
- *LOG_DEADLOCKS* Option on page 570
- *LOGIN_MODE* Option on page 571
- *LOGIN_PROCEDURE* Option on page 572
- *MAIN_RESERVED_DBSPACE_MB* Option on page 572
- *MAX_CARTESIAN_RESULT* Option on page 573
- *MAX_CLIENT_NUMERIC_PRECISION* Option on page 574
- *MAX_CLIENT_NUMERIC_SCALE* Option on page 575
- *MAX_CUBE_RESULT* Option on page 576
- *MAX_CURSOR_COUNT* Option on page 576
- *MAX_HASH_ROWS* Option on page 577
- *MAX_IQ_THREADS_PER_CONNECTION* Option on page 578
- *MAX_IQ_THREADS_PER_TEAM* Option on page 578

- *MAX_JOIN_ENUMERATION* Option on page 579
- *MAX_PARTITIONED_HASH_MB* Option on page 580
- *MAX_PREFIX_PER_CONTAINS_PHRASE* Option on page 581
- *MAX_QUERY_PARALLELISM* Option on page 581
- *MAX_QUERY_TIME* Option on page 582
- *MAX_STATEMENT_COUNT* Option on page 583
- *MAX_TEMP_SPACE_PER_CONNECTION* Option on page 583
- *MINIMIZE_STORAGE* Option on page 585
- *MIN_PASSWORD_LENGTH* Option on page 586
- *MIN_ROLE_ADMINS* Option on page 586
- *MONITOR_OUTPUT_DIRECTORY* Option on page 587
- *MPX_AUTOEXCLUDE_TIMEOUT* Option on page 588
- *MPX_HEARTBEAT_FREQUENCY* Option on page 588
- *MPX_IDLE_CONNECTION_TIMEOUT* Option on page 589
- *MPX_LIVENESS_TIMEOUT* Option on page 589
- *MPX_MAX_CONNECTION_POOL_SIZE* Option on page 590
- *MPX_MAX_UNUSED_POOL_SIZE* Option on page 591
- *MPX_WORK_UNIT_TIMEOUT* Option on page 591
- *NOEXEC* Option on page 593
- *NON_ANSI_NULL_VARCHAR* Option on page 593
- *NOTIFY_MODULUS* Option on page 595
- *ODBC_DISTINGUISH_CHAR_AND_VARCHAR* Option on page 595
- *ON_CHARSET_CONVERSION_FAILURE* Option on page 596
- *OS_FILE_CACHE_BUFFERING* Option on page 599
- *OS_FILE_CACHE_BUFFERING_TEMPDB* Option on page 600
- *POST_LOGIN_PROCEDURE* Option on page 601
- *PRECISION* Option on page 602
- *PREFETCH* Option on page 603
- *PREFETCH_BUFFER_LIMIT* Option on page 603
- *PREFETCH_BUFFER_PERCENT* Option on page 604
- *PREFETCH_GARRAY_PERCENT* Option on page 604
- *PREFETCH_SORT_PERCENT* Option on page 605
- *PRESERVE_SOURCE_FORMAT* Option [database] on page 606
- *QUERY_DETAIL* Option on page 606
- *QUERY_NAME* Option on page 607
- *QUERY_PLAN* Option on page 608
- *QUERY_PLAN_AFTER_RUN* Option on page 608
- *QUERY_PLAN_AS_HTML* Option on page 609
- *QUERY_PLAN_AS_HTML_DIRECTORY* Option on page 611

- *QUERY_PLAN_MIN_TIME* Option on page 612
- *QUERY_PLAN_TEXT_ACCESS* Option on page 613
- *QUERY_PLAN_TEXT_CACHING* Option on page 614
- *QUERY_ROWS_RETURNED_LIMIT* Option on page 615
- *QUERY_TEMP_SPACE_LIMIT* Option on page 615
- *QUERY_TIMING* Option on page 616
- *RECOVERY_TIME* Option on page 618
- *RESERVED_KEYWORDS* Option on page 618
- *RETURN_DATE_TIME_AS_STRING* Option on page 619
- *REVERT_TO_V15_OPTIMIZER* Option on page 620
- *ROUND_TO_EVEN* Option on page 620
- *ROW_COUNT* Option on page 621
- *RV_AUTO_MERGE_EVAL_INTERVAL* Option on page 622
- *RV_MERGE_NODE_MEMSIZE* Option on page 623
- *RV_MERGE_TABLE_MEMPERCENT* Option on page 624
- *RV_MERGE_TABLE_NUMROWS* Option on page 625
- *RV_RESERVED_DBSPACE_MB* Option on page 625
- *SCALE* Option on page 626
- *SNAPSHOT_VERSIONING* Option on page 627
- *SIGNIFICANTDIGITSFORDOUBLEEQUALITY* Option on page 628
- *SORT_COLLATION* Option on page 628
- *SORT_PINNABLE_CACHE_PERCENT* Option on page 629
- *SUBQUERY_CACHING_PREFERENCE* Option on page 633
- *SUBQUERY_FLATTENING_PERCENT* Option on page 634
- *SUBQUERY_FLATTENING_PREFERENCE* Option on page 635
- *SUBQUERY_PLACEMENT_PREFERENCE* Option on page 636
- *SUPPRESS_TDS_DEBUGGING* Option on page 637
- *SWEEPER_THREADS_PERCENT* Option on page 638
- *TDS_EMPTY_STRING_IS_NULL* Option [database] on page 638
- *TEMP_EXTRACT_APPEND* Option on page 639
- *TEMP_EXTRACT_BINARY* Option on page 640
- *TEMP_EXTRACT_COLUMN_DELIMITER* Option on page 641
- *TEMP_EXTRACT_DIRECTORY* Option on page 642
- *TEMP_EXTRACT_ESCAPE_QUOTES* Option on page 643
- *TEMP_EXTRACT_NAME* Options on page 643
- *TEMP_EXTRACT_NULL_AS_EMPTY* Option on page 645
- *TEMP_EXTRACT_NULL_AS_ZERO* Option on page 646
- *TEMP_EXTRACT_QUOTE* Option on page 647
- *TEMP_EXTRACT_QUOTES* Option on page 648

Database Options

- *TEMP_EXTRACT_QUOTES_ALL Option* on page 648
- *TEMP_EXTRACT_ROW_DELIMITER Option* on page 649
- *TEMP_EXTRACT_SIZE Options* on page 650
- *TEMP_EXTRACT_SWAP Option* on page 651
- *TEMP_RESERVED_DBSPACE_MB Option* on page 652
- *TEMP_SPACE_LIMIT_CHECK Option* on page 652
- *TEXT_DELETE_METHOD Option* on page 653
- *TIME_FORMAT Option* on page 654
- *TIMESTAMP_FORMAT Option* on page 655
- *TOP_NSORT_CUTOFF_PAGES Option* on page 656
- *TRIM_PARTIAL_MBC Option* on page 657
- *TRUSTED_CERTIFICATES_FILE Option* on page 658
- *USER_RESOURCE_RESERVATION Option* on page 659
- *VERIFY_PASSWORD_FUNCTION Option* on page 659
- *WASH_AREA_BUFFERS_PERCENT Option* on page 662
- *WAIT_FOR_COMMIT Option* on page 663
- *WD_DELETE_METHOD Option* on page 663

Data Extraction Options

The data extraction facility allows you to extract data from a database by redirecting the output of a **SELECT** statement from the standard interface to one or more disk files or named pipes.

The `TEMP_EXTRACT_ . . .` database options are used to control the data extraction feature.

Transact-SQL Compatibility Options

Transact-SQL compatibility options allow SAP Sybase IQ behavior to be compatible with Adaptive Server Enterprise, or to both support old behavior and allow ISO SQL92 behavior.

For further compatibility with Adaptive Server Enterprise, you can set some of these options for the duration of the current connection using the Transact-SQL **SET** statement instead of the SAP Sybase IQ **SET OPTION** statement.

See also

- *General Database Options* on page 469
- *Interactive SQL Options* on page 476
- *Alphabetical List of Options* on page 477
- *SET Statement [T-SQL]* on page 427
- *ALLOW_NULLS_BY_DEFAULT Option [TSQL]* on page 479
- *ANSI_CLOSE_CURSORS_ON_ROLLBACK Option [TSQL]* on page 480
- *ANSI_PERMISSIONS Option [TSQL]* on page 481

- *ANSINULL Option [TSQL]* on page 482
- *ANSI_SUBSTRING Option [TSQL]* on page 483
- *CHAINED Option [TSQL]* on page 495
- *CLOSE_ON_ENDTRANS Option [TSQL]* on page 497
- *CONTINUE_AFTER_RAISERROR Option [TSQL]* on page 498
- *CONVERSION_ERROR Option [TSQL]* on page 499
- *DIVIDE_BY_ZERO_ERROR Option [TSQL]* on page 526
- *NEAREST_CENTURY Option [TSQL]* on page 592
- *NON_KEYWORDS Option [TSQL]* on page 594
- *ON_ERROR Option [Interactive SQL]* on page 597
- *ON_TSQL_ERROR Option [TSQL]* on page 598
- *QUOTED_IDENTIFIER Option [TSQL]* on page 617
- *SQL_FLAGGER_ERROR_LEVEL Option [TSQL]* on page 630
- *SQL_FLAGGER_WARNING_LEVEL Option [TSQL]* on page 631
- *STRING_RTRUNCATION Option [TSQL]* on page 632
- *TSQL_VARIABLES Option [TSQL]* on page 658

Transact-SQL Option Settings for Adaptive Server Enterprise Compatibility

The default setting for some options differs from the Adaptive Server Enterprise default setting. To ensure compatible behavior, you should explicitly set the options.

When a connection is made using the Open Client or JDBC interfaces, some option settings are explicitly set for the current connection to be compatible with Adaptive Server Enterprise.

Table 11. Transact-SQL Options Set Explicitly for ASE Compatibility

Option	ASE-compatible setting
ALLOW_NULLS_BY_DEFAULT	OFF
ANSINULL	OFF
CHAINED	OFF
CONTINUE_AFTER_RAISERROR	ON
DATE_FORMAT	YYYY-MM-DD
DATE_ORDER	MDY
ESCAPE_CHARACTER	OFF
ISOLATION_LEVEL	1
ON_TSQL_ERROR	CONDITIONAL

Option	ASE-compatible setting
QUOTED_IDENTIFIER	OFF
TIME_FORMAT	HH:NN:SS.SSS
TIMESTAMP_FORMAT	YYYY-MM-DD HH:NN:SS.SSS
TSQL_VARIABLES	OFF

Interactive SQL Options

Interactive SQL options change how Interactive SQL interacts with the database.

Syntax 1

```
SET [ TEMPORARY ] OPTION
... [ userid. | PUBLIC. ] option-name = [ option-value ]
```

Syntax 2

```
SET PERMANENT
```

Syntax 3

```
SET
```

Parameters

userid:
identifier, string or host-variable

option-name:
identifier, string or host-variable

option-value:
host-variable (indicator allowed), string, identifier,
or number

Description

Syntax 1 with the **TEMPORARY** keyword cannot be used between the **BEGIN** and **END** keywords of a compound statement.

Syntax 2 **SET PERMANENT** stores all current Interactive SQL options in the `SYSOPTIONS` system table. These settings are automatically established every time Interactive SQL is started for the current user ID.

Syntax 3 is used to display all of the current option settings. If there are temporary options set for Interactive SQL or the database server, these are displayed; otherwise, the permanent option settings are displayed.

See also

- *General Database Options* on page 469
- *Transact-SQL Compatibility Options* on page 474
- *Alphabetical List of Options* on page 477
- *DEFAULT_ISQL_ENCODING Option [Interactive SQL]* on page 520
- *ON_ERROR Option [Interactive SQL]* on page 597

Alphabetical List of Options

Descriptions of general, Transact-SQL compatibility, and Interactive SQL database options. Some option names are followed by a class indicator in square brackets.

The database option class indicators are:

- [Interactive SQL] – The option changes how Interactive SQL interacts with the database.
- [TSQL] – The option allows SAP Sybase IQ behavior to be made compatible with Adaptive Server Enterprise, or to both support old behavior and allow ISO SQL92 behavior.

See also

- *Introduction to Database Options* on page 463
- *General Database Options* on page 469
- *Transact-SQL Compatibility Options* on page 474
- *Interactive SQL Options* on page 476

AFFINITY_AUTOEXCLUDE_TIMEOUT Option

The amount of time before SAP Sybase IQ removes a shut down node from the affinity map and reassigns its partitions to other nodes.

Allowed Values

0 to 10080 minutes (1 week)

Default

10 minutes

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

The amount of time before SAP Sybase IQ removes a shut down node from the affinity map and reassigns its partitions to other nodes.

AGGREGATION_PREFERENCE Option

Controls the choice of algorithms for processing an aggregate.

Allowed Values

-6 to 6

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

For aggregation (**GROUP BY, DISTINCT, SET** functions) within a query, the SAP Sybase IQ optimizer has a choice of several algorithms for processing the aggregate.

AGGREGATION_PREFERENCE lets you override the costing decision of the optimizer when choosing the algorithm. the option does not override internal rules that determine whether an algorithm is legal within the query engine.

This option is normally used for internal testing and for manually tuning queries that the optimizer does not handle well. Only experienced DBAs should use it. Inform SAP Sybase Technical Support, if you need to set AGGREGATION_PREFERENCE, as setting this option might mean that a change to the optimizer may be appropriate.

Value	Action
0	Let the optimizer choose
1	Prefer aggregation with a sort
2	Prefer aggregation using IQ indexes
3	Prefer aggregation with a hash
4	Prefer aggregation with a distinct/grouping sort

Value	Action
5	Prefer aggregation with a sort if grouping columns include all the partitioning keys of a hash partitioned table.
6	Prefer aggregation with a hash if grouping columns include all the partitioning keys of a hash partitioned table.
-1	Avoid aggregation with a sort
-2	Avoid aggregation using IQ indexes
-3	Avoid aggregation with a hash
-4n	Avoid aggregation with a distinct/grouping sort
-5	Avoid aggregation with a sort if grouping columns include all the partitioning keys of a hash partitioned table.
-6	Avoid aggregation with a hash if grouping columns include all the partitioning keys of a hash partitioned table.

ALLOW_NULLS_BY_DEFAULT Option [TSQL]

Controls whether new columns created without specifying either NULL or NOT NULL are allowed to contain NULL values.

Allowed values

ON, OFF

Default

ON

OFF for Open Client and JDBC connections

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The ALLOW_NULLS_BY_DEFAULT option is included for Transact-SQL compatibility.

ALLOW_SNAPSHOT_VERSIONING Option

Applies to all base tables in the database (as opposed to RLV-enabled tables only). Restricts table versioning for all base tables to either table-level or row-level snapshot versioning. This option does not apply to the IQ catalog store.

Allowed Values

any, table-level, row-level

Default

any

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

Value	Action
any	No restrictions on snapshot versioning.
row-level	Allows only row-level snapshot versioning. Any transactions attempting to use table-level versioning to modify a table will fail with an Illegal snapshot isolation error.
table-level	Allows only table-level snapshot versioning. Any transactions attempting to use row-level versioning to modify a table will fail with an Illegal snapshot isolation error.

ANSI_CLOSE_CURSORS_ON_ROLLBACK Option [TSQL]

Controls whether cursors that were opened **WITH HOLD** are closed when a **ROLLBACK** is performed.

Allowed Values

ON

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The ANSI SQL/3 standard requires all cursors be closed when a transaction is rolled back. This option forces that behavior and cannot be changed. The CLOSE_ON_ENDTRANS option overrides this option.

ANSI_PERMISSIONS Option [TSQL]

Controls permissions checking for DELETE and UPDATE statements.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

With ANSI_PERMISSIONS ON, SQL92 permissions requirements for **DELETE** and **UPDATE** statements are checked. The default value is OFF in Adaptive Server Enterprise. This table outlines the differences:

Table 12. Effect of ANSI_PERMISSIONS Option

SQL statement	Permissions required with ANSI_PERMISSIONS OFF	Permissions required with ANSI_PERMISSIONS ON
UPDATE	UPDATE permission on the columns where values are being set	UPDATE permission on the columns where values are being set SELECT permission on all columns appearing in the WHERE clause. SELECT permission on all columns on the right side of the set clause.
DELETE	DELETE permission on table	DELETE permission on table. SELECT permission on all columns appearing in the WHERE clause.

ANSINULL Option [TSQL]

Controls the interpretation of using = and != with NULL.

Allowed Values

ON, OFF

Default

ON

OFF for OPen CLient and JDBC connections

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

With ANSINULL ON, results of comparisons with NULL using '=' or '!=' are unknown. This includes results of comparisons implied by other operations such as CASE.

Setting ANSINULL to OFF allows comparisons with NULL to yield results that are not unknown, for compatibility with Adaptive Server Enterprise.

Note: Unlike SQL Anywhere, SAP Sybase IQ does not generate the warning “null value eliminated in aggregate function” (SQLSTATE=01003) for aggregate functions on columns containing NULL values.

ANSI_SUBSTRING Option [TSQL]

Controls the behavior of the SUBSTRING (SUBSTR) function when negative values are provided for the start or length parameters.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When the ANSI_SUBSTRING option is set to ON, the behavior of the **SUBSTRING** function corresponds to ANSI/ISO SQL/2003 behavior. A negative or zero start offset is treated as if the string were padded on the left with noncharacters, and gives an error if a negative length is provided.

When this option is set to OFF, the behavior of the **SUBSTRING** function is the same as in earlier versions of SAP Sybase IQ: a negative start offset means an offset from the end of the string, and a negative length means the desired substring ends length characters to the left of the starting offset. Using a start offset of 0 is equivalent to a start offset of 1.

Avoid using nonpositive start offsets or negative lengths with the **SUBSTRING** function. Where possible, use the **LEFT** or **RIGHT** functions instead.

Example

These examples show the difference in the values returned by the **SUBSTRING** function based on the setting of the ANSI_SUBSTRING option:

```
SUBSTRING( ' abcdefgh', -2, 4 );
ansi_substring = Off ==> 'gh'
// substring starts at second-last character
ansi_substring = On  ==> 'gh'
```

Database Options

```
// takes the first 4 characters of
// ???abcdefgh and discards all ?

SUBSTRING( 'abcdefgh',4,-2 );
ansi_substring = Off ==> 'cd'
ansi_substring = On  ==> value -2 out of range
for destination

SUBSTRING( 'abcdefgh',0,4 );
ansi_substring = Off ==> 'abcd'
ansi_substring = On  ==> 'abcd'
```

ANSI_UPDATE_CONSTRAINTS Option

Controls the range of updates that are permitted.

Allowed Values

OFF, CURSORS, STRICT

Default

CURSORS

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

SAP Sybase IQ provides several extensions that allow updates that are not permitted by the ANSI SQL standard. These extensions provide powerful, efficient mechanisms for performing updates. However, in some cases, they cause behavior that is not intuitive. This behavior might produce anomalies such as lost updates if the user application is not designed to expect the behavior of these extensions.

ANSI_UPDATE_CONSTRAINTS controls whether updates are restricted to those permitted by the SQL92 standard.

If the option is set to STRICT, these updates are prevented:

- Updates of cursors containing JOINS
- Updates of columns that appear in an **ORDER BY** clause
- The **FROM** clause is not allowed in **UPDATE** statements.

If the option is set to **CURSOR**S, these same restrictions are in place, but only for cursors. If a cursor is not opened with **FOR UPDATE** or **FOR READ ONLY**, the database server determines whether updates are permitted based on the SQL92 standard.

If **ANSI_UPDATE_CONSTRAINTS** is set to **CURSOR**S or **STRICT**, cursors containing an **ORDER BY** clause default to **FOR READ ONLY**; otherwise, they continue to default to **FOR UPDATE**.

Example

This code has a different effect, depending on the setting of **ANSI_UPDATE_CONSTRAINTS**:

```
CREATE TABLE mmg (a CHAR(3));
CREATE TABLE mmg1 (b CHAR(3));

INSERT INTO mmg VALUES ('001');
INSERT INTO mmg VALUES ('002');
INSERT INTO mmg VALUES ('003');
INSERT INTO mmg1 VALUES ('003');
SELECT * FROM mmg;
SELECT * FROM mmg1;
```

Option 1: Set ANSI_UPDATE_CONSTRAINTS to STRICT:

```
SET OPTION public.Ansi_update_constraints = 'strict';
DELETE MMG FROM MMG1 WHERE A=B;
```

This results in an error indicating that the attempted update operation is not allowed.

Option 2: Set ANSI_UPDATE_CONSTRAINTS to CURSORS or **OFF**:

```
SET OPTION public.Ansi_update_constraints = 'CURSOR'; // or 'OFF'
DELETE mmg FROM mmg1 WHERE A=B;
```

In this case, the deletion should complete without the error.

See also

- *UPDATE Statement* on page 448

ALLOW_READ_CLIENT_FILE Option

Enables client-side data transfer.

Allowed values

On, Off

Default

Off

Scope

Option can be set at the database (**PUBLIC**) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

Database Options

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Remarks

Enable this option to read from files on a client computer, for example by using the **READ_CLIENT_FILE** function.

ASE_BINARY_DISPLAY Option

Specifies that the display of SAP Sybase IQ binary columns is consistent with the display of Adaptive Server Enterprise binary columns.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

ASE_BINARY_DISPLAY affects the output of the **SELECT** statement.

This option affects only columns in the IQ store. It does not affect variables, catalog store columns or SQL Anywhere columns. When this option is ON, SAP Sybase IQ displays the column in readable ASCII format; for example, 0x1234567890abcdef. When this option is OFF, SAP Sybase IQ displays the column as binary output (not ASCII).

Set **ASE_BINARY_DISPLAY** OFF to support bulk copy operations on binary data types. SAP Sybase IQ supports bulk loading of remote data via the **LOAD TABLE USING CLIENT FILE** statement.

See also

- *LOAD TABLE Statement* on page 343

ASE_FUNCTION_BEHAVIOR Option

Specifies that output of SAP Sybase IQ functions, including **INTTOHEX** and **HEXTOINT**, is consistent with the output of Adaptive Server Enterprise functions.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When **ASE_FUNCTION_BEHAVIOR** is ON, some of the SAP Sybase IQ data type conversion functions, including **HEXTOINT** and **INTTOHEX**, return output that is consistent with the output of Adaptive Server Enterprise functions. The differences in the ASE and SAP Sybase IQ output, with respect to formatting and length, exist because ASE primarily uses signed 32-bit as the default and SAP Sybase IQ primarily uses unsigned 64-bit as the default.

SAP Sybase IQ does not provide support for 64-bit integer, as ASE does not have a 64-bit integer data type.

Example

In this example, the **HEXTOINT** function returns a different value based on whether **ASE_FUNCTION_BEHAVIOR** is ON or OFF.

The **HEXTOINT** function returns 4294967287 with **ASE_FUNCTION_BEHAVIOR** OFF:

```
select hextoint('ffffffff7') from iq_dummy
```

The **HEXTOINT** function returns -9 with **ASE_FUNCTION_BEHAVIOR** ON:

```
select hextoint('ffffffff7') from iq_dummy
```

See also

- *CONVERSION_ERROR Option [TSQL]* on page 499

AUDITING Option [database]

Enables and disables auditing in the database.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description

Auditing is the recording of details about many events in the database in the transaction log. Auditing provides some security features, at the cost of some performance. When you turn on auditing for a database, you cannot stop using the transaction log. You must turn auditing off before you turn off the transaction log. Databases with auditing on cannot be started in read-only mode.

For the AUDITING option to work, you must set the auditing option to ON, and use the **sa_enable_auditing_type** system procedure to indicate the types of information to audit, including any combination of permission checks, connection attempts, DDL statements, public options, triggers. Auditing will not take place if:

- The AUDITING option is set to OFF, or
- Auditing options have been disabled.

If you set the AUDITING option to ON, and do not specify auditing options, all types of auditing information are recorded.

BASE_TABLES_IN_RLV_STORE Option

Registers newly created tables in the RLV store, enabling row-level versioning. RLV-enabled tables are eligible for multiple writer concurrent access. You can override this setting at the table level using the **CREATE_TABLE** statement.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

When set to ON, newly created tables are registered in the RLV store. RLV-enabled tables are optimized for real-time updates.

The { **ENABLE | DISABLE** } **RLV STORE** clause of the **CREATE_TABLE** statement always overrides the **BASE_TABLES_IN_RLV_STORE** option.

Once **Base_Tables_in_RLV_STORE option** is enabled, any newly created IQ base tables are automatically RLV enabled. Enabling this option has no impact on existing IQ base tables.

BIT_VECTOR_PINNABLE_CACHE_PERCENT Option

Maximum percentage of a user's temp memory that a persistent bit-vector object can pin.

Allowed Values

0 – 100

Default

40

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

BIT_VECTOR_PINNABLE_CACHE_PERCENT controls the percentage of a user's temp memory allocation that any one persistent bit-vector object can pin in memory. It defaults to 40%, and should not generally be changed by users.

This option is primarily for use by Sybase Technical Support. If you change the value of **BIT_VECTOR_PINNABLE_CACHE_PERCENT**, do so with extreme caution; first analyze the effect on a wide variety of queries.

See also

- *HASH_PINNABLE_CACHE_PERCENT Option* on page 547

- *`SORT_PINNABLE_CACHE_PERCENT` Option* on page 629

BLOCKING Option

Controls the behavior in response to locking conflicts. `BLOCKING` is not supported on secondary nodes of a multiplex.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY PUBLIC OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

When `BLOCKING` is off, a transaction receives an error when it attempts a write operation and is blocked by the read lock of another transaction.

When `BLOCKING` is on, any transaction attempting to obtain a lock that conflicts with an existing lock held by another transaction waits until every conflicting lock is released or until the `blocking_timeout` is reached. If the lock is not released within `blocking_timeout` milliseconds, then an error is returned for the waiting transaction.

See also

- *`BLOCKING_TIMEOUT` Option* on page 490

BLOCKING_TIMEOUT Option

Controls the length of time a transaction waits to obtain a lock. `BLOCKING_TIMEOUT` is not supported on secondary nodes of a multiplex.

Allowed Values

Integer, in milliseconds.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When the blocking option is on, any transaction attempting to obtain a lock that conflicts with an existing lock waits for the indicated number of milliseconds for the conflicting lock to be released. If the lock is not released within blocking_timeout milliseconds, an error is returned for the waiting transaction.

Set the option to 0 to force all transactions attempting to obtain a lock to wait until all conflicting transactions release their locks.

See also

- *BLOCKING Option* on page 490

BT_PREFETCH_MAX_MISS Option

Controls the way SAP Sybase IQ determines whether to continue prefetching B-tree pages for a given query.

Allowed Values

0 – 1000

Default

2

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Use only if instructed to do so by SAP Sybase Technical Support. For queries that use **HG** (High_Group) indexes, SAP Sybase IQ prefetches B-tree pages sequentially until it

Database Options

determines that prefetching is no longer useful. For some queries, it might turn off prefetching prematurely. Increasing the value of `BT_PREFETCH_MAX_MISS` makes it more likely that SAP Sybase IQ continues prefetching, but also might increase I/O unnecessarily.

If queries using **HG** indexes run more slowly than expected, try gradually increasing the value of `BT_PREFETCH_MAX_MISS`.

Experiment with different settings to find the setting that gives the best performance. For most queries, useful settings are in the range of 1 to 10.

See also

- *BT_PREFETCH_SIZE Option* on page 492
- *PREFETCH_BUFFER_LIMIT Option* on page 603

BT_PREFETCH_SIZE Option

Restricts the size of the read-ahead buffer for the High_Group B-tree.

Allowed Values

0 – 100. Setting to 0 disables B-tree prefetch.

Default

10

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

B-tree prefetch is activated by default for any sequential access to the High_Group index such as **INSERT**, large **DELETE**, range predicates, and DBCC (Database Consistency Checker) commands.

`BT_PREFETCH_SIZE` limits the size of the read-ahead buffer for B-tree pages. Reducing prefetch size frees buffers, but also degrades performance at some point. Increasing prefetch size might have marginal returns. This option should be used in conjunction with the options `PREFETCH_GARRAY_PERCENT`, `GARRAY_INSERT_PREFETCH_SIZE`, and `GARRAY_RO_PREFETCH_SIZE` for non-unique High_Group indexes.

See also

- *GARRAY_INSERT_PREFETCH_SIZE Option* on page 545
- *GARRAY_RO_PREFETCH_SIZE Option* on page 546
- *PREFETCH_GARRAY_PERCENT Option* on page 604

BTREE_PAGE_SPLIT_PAD_PERCENT Option

Determines per-page fill factor during page splits for B-Tree structures.

Allowed Values

0 – 90

Default

50

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

B-Tree structures are used by the HG, LF, DT, TIME, and DTTM indexes. Splits of a B-Tree page try to leave the specified percentage empty to avoid splitting when new keys are inserted into the index.

Indexes reserve storage at the page level that can be allocated to new keys as additional data is inserted. Reserving space consumes additional disk space, but can help the performance of incremental inserts. If future plans include incremental inserts, and the new rows do not have values that are already present in the index, a nonzero value for GARRAY_PAGE_SPLIT_PAD_PERCENT may improve incremental insert performance.

If you do not plan to incrementally update the index, you can reduce the value of this option to save disk space.

See also

- *GARRAY_FILL_FACTOR_PERCENT Option* on page 544
- *GARRAY_PAGE_SPLIT_PAD_PERCENT Option* on page 545

CACHE_AFFINITY_PERCENT Option

The maximum percentage of main cache to use for affinity. Non-affinity data can use this area if insufficient affinity data exists.

Allowed Values

0 - 100 %

Default

70

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. The database server must be shut down and restarted for the change to take effect

Description

This option defines the percentage of the buffer cache used for affinitized data buffers. SAP Sybase IQ buffer caches are organized as a long MRU/LRU chain. Non-affinitized data buffers are put into the chain after affinitized buffers when this percentage is non-zero, so that affinitized data stay in the cache longer than non-affinitized data. If there are insufficient affinitized data buffers to fill this entire percentage, non-affinitized data may consume the remainder.

Note: Before changing this option, check the value of the WASH_AREA_BUFFERS_PERCENT option. WASH_AREA_BUFFERS_PERCENT affects the LRU side of the buffer cache and CACHE_AFFINITY_PERCENT affects the MRU side. The total of these two values cannot exceed 100 percent.

CACHE_PARTITIONS Option

Sets the number of partitions to be used for the main and temporary buffer caches.

Allowed Values

0, 1, 2, 4, 8, 16, 32, 64

- 0 – (default) SAP Sybase IQ computes the number of partitions automatically as $\text{number_of_cpus}/8$, rounded to the nearest power of 2, up to a maximum of 64.
- 1 – one partition only; this value disables partitioning.
- 2 through 64 – number of partitions; must be a power of 2.

Default

0

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect for the current database the next time you start the database server.

Description

Partitioning the buffer cache can sometimes improve performance on systems with multiple CPUs by reducing lock contention. Normally, you should rely on the value that SAP Sybase IQ calculates automatically, which is based on the number of CPUs on your system. However, if you find that load or query performance in a multi-CPU configuration is slower than expected, you might be able to improve it by setting a different value for CACHE_PARTITIONS.

Both the number of CPUs and the platform can influence the ideal number of partitions. Experiment with different values to determine the best setting for your configuration.

The value you set for CACHE_PARTITIONS applies to both the main and temp buffer caches. The absolute maximum number of partitions is 64, for each buffer cache.

The **-iqpartition start iq** server option sets the partition limit at the server level. If you specify **-iqpartition** at server startup, it overrides the CACHE_PARTITIONS setting.

The number of partitions does not affect other buffer cache settings. It also does not affect statistics collected by the IQ monitor; statistics for all partitions are rolled up and reported as a single value.

Example

In a system with 100 CPUs, if you do not set CACHE_PARTITIONS, SAP Sybase IQ automatically sets the number of partitions to 16:

$100 \text{ cpus} / 8 = 12$, rounded to 16.

With this setting, there are 16 partitions for the main cache and 16 partitions for the temp cache.

In the same system with 100 CPUs, to explicitly set the number of partitions to 8, specify:

```
SET OPTION "PUBLIC".CACHE_PARTITIONS=8
```

CHAINED Option [TSQL]

Controls transaction mode in the absence of a **BEGIN TRANSACTION** statement.

Allowed Values

ON, OFF

Default

ON

OFF for Open Client and JDBC connections

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Controls the Transact-SQL transaction mode. In unchained mode (CHAINED = OFF) each statement is committed individually unless an explicit **BEGIN TRANSACTION** statement is executed to start a transaction. In chained mode (CHAINED = ON) a transaction is implicitly started before any data retrieval or modification statement. For Adaptive Server Enterprise, the default setting is OFF.

CHECKPOINT_TIME Option

Set the maximum length of time, in minutes, that the database server runs without doing a checkpoint.

Allowed Values

Integer

Default

60

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must shut down and restart the database server for the change to take effect.

Description

This option is used with the RECOVERY_TIME option to decide when checkpoints should be done.

See also

- *RECOVERY_TIME Option* on page 618

CIS_ROWSET_SIZE Option

Sets the number of rows that are returned from remote servers for each fetch.

Allowed Values

Integer

Default

50

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option sets the ODBC **FetchArraySize** value when you are using ODBC to connect to a remote database server.

CLOSE_ON_ENDTRANS Option [TSQL]

Controls closing of cursors at the end of a transaction.

Allowed Values

ON

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When `CLOSE_ON_ENDTRANS` is set to `ON` (the default and only value allowed), cursors are closed at the end of a transaction, which is Transact-SQL compatible behavior.

CONTINUE_AFTER_RAISERROR Option [TSQL]

Controls behavior following a **RAISERROR** statement.

Allowed Values

`ON`, `OFF`

Default

`ON`

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY PUBLIC OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

The **RAISERROR** statement is used within procedures to generate an error. When `CONTINUE_AFTER_RAISERROR` is set to `OFF`, the execution of the procedure is stopped when the **RAISERROR** statement is encountered.

When `CONTINUE_AFTER_RAISERROR` is `ON`, the **RAISERROR** statement no longer signals an execution-ending error. Instead, the **RAISERROR** status code and message are stored and the most recent **RAISERROR** is returned when the procedure completes. If the procedure that caused the **RAISERROR** was called from another procedure, the **RAISERROR** is not returned until the outermost calling procedure terminates.

Intermediate **RAISERROR** statuses and codes are lost after the procedure terminates. If, at return time, an error occurs along with the **RAISERROR**, then the error information is returned and the **RAISERROR** information is lost. The application can query intermediate **RAISERROR** statuses by examining `@@error` global variable at different execution points.

The setting of `CONTINUE_AFTER_RAISERROR` is used to control behavior following a **RAISERROR** statement only if the `ON_TSQL_ERROR` option is set to `CONDITIONAL` (the default). If you set the `ON_TSQL_ERROR` option to `STOP` or `CONTINUE`, the

ON_TSQL_ERROR setting takes precedence over the CONTINUE_AFTER_RAISERROR setting.

See also

- *ON_TSQL_ERROR Option [TSQL]* on page 598

CONVERSION_ERROR Option [TSQL]

Controls reporting of data type conversion failures on fetching information from the database.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option controls whether data type conversion failures, when data is fetched from the database or inserted into the database, are reported by the database as errors (CONVERSION_ERROR set to ON), or as warnings (CONVERSION_ERROR set to OFF).

When CONVERSION_ERROR is set to ON, the SQLE_CONVERSION_ERROR error is generated.

If the option is set to OFF, the warning SQLE_CANNOT_CONVERT is produced. Each thread doing data conversion for a **LOAD** statement writes at most one warning message to the .iqmsg file.

If conversion errors are reported as warnings only, the NULL value is used in place of the value that could not be converted. In Embedded SQL, an indicator variable is set to -2 for the column or columns that cause the error.

CONVERSION_MODE Option

Restricts implicit conversion between binary data types (BINARY, VARBINARY, and LONG BINARY) and other non-binary data types (BIT, TINYINT, SMALLINT, INT, UNSIGNED

Database Options

INT, BIGINT, UNSIGNED BIGINT, CHAR, VARCHAR, and LONG VARCHAR) on various operations.

Allowed Values

0, 1

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The default value of 0 maintains implicit conversion behavior prior to version 12.7. Setting `CONVERSION_MODE` to 1 restricts implicit conversion of binary data types to any other non-binary data type on **INSERT**, **UPDATE**, and in queries. The restrict binary conversion mode also applies to **LOAD TABLE** default values and **CHECK** constraint. The use of this option prevents implicit data type conversions of encrypted data that would result in semantically meaningless operations.

Users must be specifically licensed to use the encrypted column functionality of the SAP Sybase IQ Advanced Security Option.

Implicit Conversion Restrictions

The `CONVERSION_MODE` option restrict binary mode value of 1 (`CONVERSION_MODE = 1`) restricts implicit conversion for these operations:

- **LOAD TABLE** with **CHECK** constraint or default value
- **INSERT...SELECT**, **INSERT...VALUE**, and **INSERT...LOCATION**
- Certain types of **UPDATE**
- Certain types of **INSERT** and **UPDATE** via updatable cursor
- All aspects of queries in general

Restrict Implicit Binary Conversion Mode for LOAD TABLE

The restrict implicit binary conversion mode (CONVERSION_MODE set to 1) applies to **LOAD TABLE** with **CHECK** constraint or default value.

Example

```
CREATE TABLE t3 (c1 INT,
  csi SMALLINT,
  cvb VARBINARY(2),
  CHECK (csi<cvb));
SET TEMPORARY OPTION CONVERSION_MODE = 1;
```

This request:

```
LOAD TABLE t3(c1 ',', csi ',', cvb ',')
FROM '/s1/mydata/t3.inp'
QUOTES OFF ESCAPES OFF
ROW DELIMITED BY '\n'
```

fails with the message:

```
"Invalid data type comparison in predicate
(t3.csi < t3.cvb), [-1001013] ['QFA13']"
```

Restrict Implicit Binary Conversion Mode for INSERT

The restrict implicit binary conversion mode (CONVERSION_MODE set to 1) applies to **INSERT...SELECT**, **INSERT...VALUE**, and **INSERT...LOCATION**.

Example

```
CREATE TABLE t1 (c1 INT PRIMARY KEY,
  cbt BIT NULL,
  cti TINYINT,
  csi SMALLINT,
  cin INTEGER,
  cui UNSIGNED INTEGER,
  cbi BIGINT,
  cub UNSIGNED BIGINT,
  cch CHAR(10),
  cvc VARCHAR(10),
  cbn BINARY(8),
  cvb VARBINARY(8),
  clb LONG BINARY,
  clc LONG VARCHAR);

CREATE TABLE t2 (c1 INT PRIMARY KEY,
  cbt BIT NULL,
  cti TINYINT,
  csi SMALLINT,
  cin INTEGER,
  cui UNSIGNED INTEGER,
  cbi BIGINT,
  cub UNSIGNED BIGINT,
```

Database Options

```
cch CHAR(10),
cvc VARCHAR(10),
cbn BINARY(8),
cvb VARBINARY(8),
clb LONG BINARY,
clc LONG VARCHAR);

CREATE TABLE t4 (c1 INT, cin INT DEFAULT 0x31);

SET TEMPORARY OPTION CONVERSION_MODE = 1;
```

This request:

```
INSERT INTO t1(c1, cvb) SELECT 99, cin FROM T2
WHERE c1=1
```

fails with the message:

```
"Unable to convert column 'cvb' to the requested
datatype (varbinary) from datatype (integer).
[-1013043] ['QCA43']"
```

Restrict Implicit Binary Conversion Mode for UPDATE

The restrict implicit binary conversion mode (CONVERSION_MODE set to 1) applies to certain types of **UPDATE**.

Restrict implicit binary conversion mode applies to:

- **UPDATE SET VALUE FROM** *expression* (including constant)
- **UPDATE SET VALUE FROM** *other column*
- **UPDATE SET VALUE FROM** *host variable*
- **JOIN UPDATE SET VALUE FROM** *column of other table*

Example

This request:

```
UPDATE t1 SET cbi=cbn WHERE c1=1
```

fails with the message:

```
"Unable to implicitly convert column 'cbi' to datatype
(bigint) from datatype (binary). [-1000187] ['QCB87']"
```

Restrict Implicit Binary Conversion Mode for Positioned INSERT and Positioned UPDATE via Updatable Cursor

The restrict implicit binary conversion mode (CONVERSION_MODE set to 1) applies to certain types of **INSERT** and **UPDATE** via updatable cursor.

Restrict implicit binary conversion mode applies to:

- **PUT** *cursor-name* **USING** ... *host-variable*
- Positioned **UPDATE** from another column

- Positioned **UPDATE** from a constant
- Positioned **UPDATE** from a host variable

Restrict Implicit Binary Conversion Mode for Queries

The restrict implicit binary conversion mode (`CONVERSION_MODE` set to 1) applies to all aspects of queries in general.

Comparison Operators

When `CONVERSION_MODE = 1`, the restriction applies to these operators:

- `=`, `!=`, `<`, `<=`, `>=`, `<>`, `!>`, `!<`
- **BETWEEN ... AND**
- **IN**

used in a search condition for these clauses:

- **WHERE** clause
- **HAVING** clause
- **CHECK** clause
- **ON** phrase in a join
- **IF/CASE** expression

Example

This query:

```
SELECT COUNT(*) FROM T1
WHERE cvb IN (SELECT csi FROM T2)
```

fails with the message:

```
"Invalid data type comparison in predicate
(t1.cvb IN (SELECT t1.csi ...)), [-1001013]
['QFA13']"
```

String Functions

When `CONVERSION_MODE = 1`, the restriction applies to these string functions:

- **CHAR**
- **CHAR_LENGTH**
- **DIFFERENCE**
- **LCASE**
- **LEFT**
- **LOWER**
- **LTRIM**
- **PATINDEX**
- **RIGHT**

Database Options

- **RTRIM**
- **SIMILAR**
- **SORTKEY**
- **SOUNDEX**
- **SPACE**
- **STR**
- **TRIM**
- **UCASE**
- **UPPER**

Example

This query:

```
SELECT ASCII(cvb) FROM t1 WHERE c1=1
```

fails with the message:

```
"Data exception - data type conversion is not possible. Argument to ASCII must be string, [-1009145] ['QFA2E']"
```

The following functions allow either a string argument or a binary argument. When `CONVERSION_MODE = 1`, the restriction applies to mixed type arguments, that is, one argument is string and the other argument is binary.

- **INSERTSTR**
- **LOCATE**
- **REPLACE**
- **STRING**
- **STUFF**

Example

This query:

```
SELECT STRING(cvb, cvc) FROM t1 WHERE c1=1
```

where the column `cvb` is defined as `VARBINARY` and the column `cvc` is defined as `VARCHAR`, fails

with the message:

```
"Data exception - data type conversion is not possible. Arguments to STRING must be all binary or all string, [-1009145] ['QFA2E']"
```

The restriction does not apply to these string functions:

- **BIT_LENGTH**
- **BYTE_LENGTH**

- **CHARINDEX**
- **LENGTH**
- **OCTET_LENGTH**
- **REPEAT**
- **REPLICATE**
- **SUBSTRING**

Arithmetic Operations and Functions

When `CONVERSION_MODE = 1`, the restriction applies to these operators used in arithmetic operations:

`+, -, *, /`

The restriction applies to these bitwise operators used in bitwise expressions:

`& (AND), | (OR), ^ (XOR)`

The restriction also applies to integer arguments of these functions:

- **ROUND**
- **“TRUNCATE”**
- **TRUNCNUM**

Example

This query:

```
SELECT ROUND(4.4, cvb) FROM t1 WHERE C1=1
```

fails with the message:

```
"Data exception - data type conversion is not possible. Second Argument to ROUND cannot be converted into an integer, [-1009145] ['QFA2E']"
```

Integer Argument to Various Functions

When `CONVERSION_MODE = 1`, the restriction applies to integer argument of these functions:

- **ARGN**
- **SUBSTRING**
- **DATEADD**
- **YMD**

Example

This query:

```
SELECT ARGN(cvb, csi, cti) FROM t1 WHERE c1=1
```

fails with the message:

Database Options

```
"Data exception - data type conversion is not possible. First Argument to ARGN cannot be converted to an integer, [-1009145] ['QFA2E']"
```

Analytical Functions, Aggregate Functions, and Numeric Functions

When `CONVERSION_MODE = 1`, no further restriction applies to analytical functions, aggregate functions, and numeric functions that require numeric expressions as arguments.

CONVERT_VARCHAR_TO_1242 Option

Converts pre-version 12.4.2 `VARCHAR` data to compressed format.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Takes effect when you run `sp_iqcheckdb` in any mode.

Description

Helps further compress data and improve performance, especially for databases with many variable character strings.

Set this option and then run `sp_iqcheckdb` only once, and only for `VARCHAR` columns that were created before version 12.4.2.

COOPERATIVE_COMMIT_TIMEOUT Option

Governs when a `COMMIT` entry in the transaction log is written to disk.

Allowed Values

Integer, in milliseconds

Default

250

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option only has meaning when COOPERATIVE_COMMITS is set to ON. The database server waits for the specified number of milliseconds for other connections to fill a page of the log before writing to disk. The default setting is 250 milliseconds.

See also

- *COOPERATIVE_COMMITS Option on page 507*

COOPERATIVE_COMMITS Option

Controls when commits are written to disk.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If COOPERATIVE_COMMITS is set to OFF, a **COMMIT** is written to disk as soon as the database server receives it, and the application is then allowed to continue.

If COOPERATIVE_COMMITS is set to ON, the default, the database server does not immediately write the **COMMIT** to the disk. Instead, it requires the application to wait for a maximum length set by the COOPERATIVE_COMMIT_TIMEOUT option for something else to put on the pages before the commit is written to disk.

Setting `COOPERATIVE_COMMITS` to `ON`, and increasing the `COOPERATIVE_COMMIT_TIMEOUT` setting increases overall database server throughput by cutting down the number of disk I/Os, but at the expense of a longer turnaround time for each individual connection.

See also

- *COOPERATIVE_COMMIT_TIMEOUT* Option on page 506

CREATE_HG_WITH_EXACT_DISTINCTS

Determines whether the database engine creates tiered HG or single-tiered HG indexes.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY PUBLIC OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

`CREATE_HG_WITH_EXACT_DISTINCTS` determines whether the database engine creates HG indexes as single HG or tiered HG:

- If `CREATE_HG_WITH_EXACT_DISTINCTS='ON'`, all subsequent HG indexes explicitly created with a **CREATE INDEX** command or implicitly creating or altering a table with a `PRIMARY KEY` or a `FOREIGN KEY` declaration, will be non-tiered HG indexes.
- If `CREATE_HG_WITH_EXACT_DISTINCTS='OFF'`, all subsequent HG indexes explicitly created with a **CREATE INDEX** command or implicitly creating or altering a table with a `PRIMARY KEY` or a `FOREIGN KEY` declaration, will be tiered HG.

This option is `ON` by default in all newly created 16.0 databases, and all 16.0 database upgraded from SAP Sybase IQ 15.x. To take advantage of the new tiered structure, set this option to `OFF`. Use `sp_iqrebuildindex` to convert non-tiered HG indexes to tiered HG and vice-versa.

CREATE_HG_AND_FORCE_PHYSICAL_DELETE

Governs 16.0 tiered HG index delete behavior.

Allowed Values

ON/OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

CREATE_HG_AND_FORCE_PHYSICAL_DELETE determines whether delete operation physically removes rows from an HG immediately or defers the removal to a point later in the load:

- Setting CREATE_HG_AND_FORCE_PHYSICAL_DELETE='ON' (default) instructs SAP Sybase IQ to perform a physical delete, which increases the performance of some queries (link in .. list and ordered projection, for example), but can cause delete queries on tables with tiered HG indexes to run more slowly.
- Setting CREATE_HG_AND_FORCE_PHYSICAL_DELETE='OFF' instructs SAP Sybase IQ to perform a virtual or deferred delete, which improves delete query performance, but can impact queries on tables with tiered HG indexes.

Set CREATE_HG_AND_FORCE_PHYSICAL_DELETE before creating a tiered HG column index. It does not affect preexisting HG indexes. It has no effect on **sp_iqrebuildindex**. This option persists through the life of the tiered HG index, and cannot be changed or modified unless the index is dropped and the option toggled before recreating the index (**sp_iqrebuildindex** cannot modify the status of the index).

Note: **sp_iqrebuildindex** output includes a Force Physical Delete column that identifies the status of this option.

CURSOR_WINDOW_ROWS Option

Defines the number of cursor rows to buffer.

Allowed Values

20 – 100000

Default

200

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set only for individual connections or the PUBLIC role. You must shut down and restart the database server for the change to take effect.

Description

When an application opens a cursor, SAP Sybase IQ creates a FIFO (first-in, first-out) buffer to hold the data rows generated by the query. CURSOR_WINDOW_ROWS defines how many rows can be put in the buffer. If the cursor is opened in any mode other than NO SCROLL, SAP Sybase IQ allows for backward scrolling for up to the total number of rows allowed in the buffer before it must restart the query. This is not true for NO SCROLL cursors, as they do not allow backward scrolling.

For example, with the default value for this option, the buffer initially holds rows 1 through 200 of the query result set. If you fetch the first 300 rows, the buffer holds rows 101 through 300. You can scroll backward or forward within that buffer with very little overhead cost. If you scroll before row 101, SAP Sybase IQ restarts that query until the required row is back in the buffer. This can be an expensive operation to perform, so your application should avoid it where possible. An alternative is to increase the value for CURSOR_WINDOW_ROWS to accommodate a larger possible scrolling area; however, the default setting of 200 is sufficient for most applications.

DATE_FIRST_DAY_OF_WEEK Option

Determines the first day of the week.

Allowed Values

0 – 6

Default

0 (Sunday)

Scope

Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role. Can be set for individual connections or for PUBLIC role.

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set for an individual connection or for the PUBLIC role. You must shut down and restart the database server for the change to take effect.

Description

This option specifies which day is the first day of the week. By default, Sunday is day 1, Monday is day 2, Tuesday is day 3, and so on:

Table 13. DATE_FIRST_DAY_OF_WEEK Valid Values

Value	First Day
0	Sunday
1	Monday
2	Tuesday
3	Wednesday
4	Thursday
5	Friday
6	Saturday

For example, if you change the value of DATE_FIRST_DAY_OF_WEEK to 3, Wednesday becomes day 1, Thursday becomes day 2, and so on. This option only affects the **DOW** and **DATEPART** functions.

The SQL Anywhere option FIRST_DAY_OF_WEEK performs the same function, but assigns the values 1 through 7 instead of 0 through 6. 1 stands for Monday and 7 for Sunday (the default).

DATE_FORMAT Option

Sets the format used for dates retrieved from the database.

Allowed Values

String

Default

'YYYY-MM-DD'. This corresponds to ISO date format specifications.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The format is a string using these symbols:

Table 14. Symbols Used in DATE_FORMAT String

Symbol	Description
yy	2-digit year
yyyy	4-digit year
mm	2-digit month, or 2-digit minutes if following a colon (as in 'hh:mm')
mmm	3-character name of month
mmmm[m...]	Character long form for months—as many characters as there are m's, until the number of m's specified exceeds the number of characters in the month's name.
d	Single-digit day of week, (0 = Sunday, 6 = Saturday)
dd	2-digit day of month
ddd	3-character name of the day of week.
dddd[d...]	Character long form for day of the week—as many characters as there are d's, until the number of d's specified exceeds the number of characters in the day's name.
jjj	Day of the year, from 1 to 366

Note: Multibyte characters are not supported in date format strings. Only single-byte characters are allowed, even when the collation order of the database is a multibyte collation order like 932JPN. Use the concatenation operator to include multibyte characters in date format strings. For example, if '?' represents a multibyte character, use the concatenation operator to move the multibyte character outside of the date format string:

```
SELECT DATEFORMAT (StartDate, 'yy') + '?'
FROM Employees;
```

Each symbol is substituted with the appropriate data for the date being formatted. Any format symbol that represents character rather than digit output can be put in uppercase which causes the substituted characters to also be in uppercase. For numbers, using mixed case in the format string suppresses leading zeros.

You can control the padding of numbers by changing the case of the symbols. Same-case values (MM, mm, DD, or dd) all pad number with zeros. Mixed-case (Mm, mM, Dd, or dD) cause the number to not be zero-padded; the value takes as much room as required. For example:

```
SELECT dateformat ( cast ('2011/01/01' as date ), 'yyyy/Mm/Dd' )
```

returns this value:

```
2011/1/1
```

Examples

This table illustrates `DATE_FORMAT` settings, together with the output from this statement, executed on Saturday May 21, 2011:

```
SELECT CURRENT DATE
```

Table 15. DATE_FORMAT Settings

DATE_FORMAT	SELECT CURRENT DATE
yyyy/mm/dd/ddd	2011/05/21/sat
jjj	141
mmm yyyy	may 2011
mm-yyyy	05-2011

See also

- *RETURN_DATE_TIME_AS_STRING* Option on page 619
- *TIME_FORMAT* Option on page 654

DATE_ORDER Option

Controls the interpretation of date formats.

Allowed Values

'MDY', 'YMD', or 'DMY'

Default

'YMD'. This corresponds to ISO date format specifications.

'MDY' for OPne Client and DJBC connections

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

DATE_ORDER is used to determine whether 10/11/12 is Oct 11 1912, Nov 12 1910, or Nov 10 1912. The option can have the value 'MDY', 'YMD', or 'DMY'.

DBCC_LOG_PROGRESS Option

Reports the progress of the **sp_iqcheckdb** system stored procedure.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When `DBCC_LOG_PROGRESS` is ON, the `sp_iqcheckdb` system stored procedure sends progress messages to the IQ message file. These messages allow the user to follow the progress of the `sp_iqcheckdb` operation.

Stored procedures are documented in *Reference: Building Blocks, Tables, and Procedures*.

Examples

Sample progress log output of the command `sp_iqcheckdb 'check database'`:

```
IQ Utility Check Database
Start CHECK STATISTICS table: tloansf
Start CHECK STATISTICS for field: aqsn_dt
Start CHECK STATISTICS processing index:
IQ_IDX_T444_C1_FP
Start CHECK STATISTICS processing index:
tloansf_aqsn_dt_HNG
Done CHECK STATISTICS field: aqsn_dt
```

Sample progress log output of the command `sp_iqcheckdb 'allocation table nation'`:

```
Start ALLOCATION table: nation
Start ALLOCATION processing index: nationhg1
Done ALLOCATION table: nation
Done ALLOCATION processing index: nationhg1
```

DBCC_PINNABLE_CACHE_PERCENT Option

Controls the percent of the cache used by the `sp_iqcheckdb` system stored procedure.

Allowed Values

0 – 100

Default

50

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporarily for an individual connection or for the PUBLIC role. Takes effect at the next execution of `sp_iqcheckdb`.

Description

The **sp_iqcheckdb** system stored procedure works with a fixed number of buffers, as determined by this option. By default, a large percentage of the cache is reserved to maximize **sp_iqcheckdb** performance.

Stored procedures are documented in *Reference: Building Blocks, Tables, and Procedures*.

DEBUG_MESSAGES Option

Controls whether or not **MESSAGE** statements that include a **DEBUG ONLY** clause are executed.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option allows you to control the behavior of debugging messages in stored procedures that contain a **MESSAGE** statement with the **DEBUG ONLY** clause specified. By default, this option is set to OFF and debugging messages do not appear when the **MESSAGE** statement is executed. By setting **DEBUG_MESSAGES** to ON, you can enable the debugging messages in all stored procedures.

Note: **DEBUG ONLY** messages are inexpensive when the **DEBUG_MESSAGES** option is set to OFF, so these statements can usually be left in stored procedures on a production system. However, they should be used sparingly in locations where they would be executed frequently; otherwise, they might result in a small performance penalty.

See also

- *MESSAGE Statement* on page 366

DEDICATED_TASK Option

Dedicates a request handling task to handling requests from a single connection.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set as a temporary option only, for an individual connection or for the PUBLIC role, for the duration of the current connection.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

When the DEDICATED_TASK connection option is set to ON, a request handling task is dedicated exclusively to handling requests for the connection. By pre-establishing a connection with this option enabled, you can gather information about the state of the database server if it becomes otherwise unresponsive.

DEFAULT_DBSPACE Option

Changes the default dbspace where tables are created.

Allowed Values

String containing a dbspace name

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

DEFAULT_DBSPACE allows the administrator to set the default dbspace for a role or user or allows a user to set the user's own default dbspace.

IQ_SYSTEM_TEMP is always used for global temporary tables unless a table IN clause is used that specifies SYSTEM, in which case an SA global temporary table is created.

At database creation, the system dbspace, IQ_SYSTEM_MAIN, is created and is implied when the PUBLIC.DEFAULT_DBSPACE option setting is empty or explicitly set to IQ_SYSTEM_MAIN. Immediately after creating the database, create a second main dbspace, revoke CREATE privilege in dbspace IQ_SYSTEM_MAIN from PUBLIC, grant CREATE in dbspace for the new main dbspace to selected users or PUBLIC, and set PUBLIC.DEFAULT_DBSPACE to the new main dbspace. For example:

```
CREATE DBSPACE user_main USING FILE user_main
'user_main1' SIZE 10000;
GRANT CREATE ON user_main TO PUBLIC;
REVOKE CREATE ON IQ_SYSTEM_MAIN FROM PUBLIC;
SET OPTION PUBLIC.DEFAULT_DBSPACE = 'user_main';
```

Example

In this example, CONNECT and RESOURCE privileges on all dbspaces are granted to users usrA and usrB, and each of these users is granted CREATE privilege on a particular dbspace:

```
GRANT CONNECT, RESOURCE TO usrA, usrB
IDENTIFIED BY pwdA, pwdB;
GRANT CREATE ON dbsp1 TO usrA;
GRANT CREATE ON dbsp3 TO usrB;
SET OPTION "usrA".default_dbspace = 'dbsp1';
SET OPTION "usrB".default_dbspace = 'dbsp3';
SET OPTION "PUBLIC".default_dbspace = dbsp2;

CREATE TABLE "DBA".t1(c1 int, c2 int);
INSERT INTO t1 VALUES (1, 1);
INSERT INTO t1 VALUES (2, 2);
COMMIT;
```

UsrA connects:

```
CREATE TABLE "UsrA".t1(c1 int, c2 int);
INSERT INTO t1 VALUES (1, 1);
INSERT INTO t1 VALUES (2, 2);
COMMIT;
```

UsrB connects:

```
CREATE TABLE "UsrB".t1(c1 int, c2 int);
INSERT INTO t1 VALUES (1, 1);
INSERT INTO t1 VALUES (2, 2);
COMMIT;
```


DBA connects:

```
SELECT Object, DbspaceName, ObjSize
FROM sp_iqindexinfo();
```

sp_iqindexinfo result:

DBA.t1	dbbsp2	200k
DBA.t1.ASIQ_IDX_T730_C1_FP	dbbsp2	288k
DBA.t1.ASIQ_IDX_T730_C2_FP	dbbsp2	288k
usrA.t1	dbbsp1	200k
usrA.t1.ASIQ_IDX_T731_C1_FP	dbbsp1	288k
usrA.t1.ASIQ_IDX_T731_C2_FP	dbbsp1	288k
usrB.t1	dbbsp3	200k
usrB.t1.ASIQ_IDX_T732_C1_FP	dbbsp3	288k
usrB.t1.ASIQ_IDX_T732_C2_FP	dbbsp3	288k

DEFAULT_DISK_STRIPING Option

Sets the default disk striping value for all dbspaces.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

By default, disk striping is ON for all dbspaces in the IQ main store. This option is used only by **CREATE DBSPACE** and defines the default striping value, if **CREATE DBSPACE** does not specify striping.

See also

- *CREATE DBSPACE Statement* on page 117

DEFAULT_HAVING_SELECTIVITY_PPM Option

Provides default selectivity estimates to the optimizer for most **HAVING** clauses in parts per million.

Allowed Values

0 – 1000000

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

DEFAULT_HAVING_SELECTIVITY_PPM sets the selectivity for **HAVING** clauses, overriding optimizer estimates. A **HAVING** clause filters the results of a **GROUP BY** clause or a query with a select list consisting solely of aggregate functions. When DEFAULT_HAVING_SELECTIVITY_PPM is set to the default of 0, the optimizer estimates how many rows are filtered by the **HAVING** clause. Sometimes the IQ optimizer does not have sufficient information to choose an accurate selectivity, and in these cases chooses a generic estimate of 40%. DEFAULT_HAVING_SELECTIVITY_PPM allows a user to replace the optimizer estimate for all **HAVING** predicates in a query.

Users can also specify the selectivity of individual **HAVING** clauses in the query, as described in *Reference: Building Blocks, Tables, and Procedures*.

DEFAULT_ISQL_ENCODING Option [Interactive SQL]

Specifies the code page used by **READ** and **OUTPUT** statements.

Allowed Values

identifier or *string*

Default

Use system code page (empty string)

Scope

Can only be set as a temporary option, for the duration of the current connection.

Description

DEFAULT_ISQL_ENCODING is used to specify the code page to use when reading or writing files. It cannot be set permanently. The default code page is the default code page for the platform you are running on. On English Windows machines, the default code page is 1252.

Interactive SQL determines the code page that is used for a particular **OUTPUT** or **READ** statement as follows, where code page values occurring earlier in the list take precedence over those occurring later in the list:

- The code page specified in the **ENCODING** clause of the **OUTPUT** or **READ** statement
- The code page specified with the `DEFAULT_ISQL_ENCODING` option (if this option is set)
- The default code page for the computer on which Interactive SQL is running

Example

Set the encoding to UTF-16 (for reading Unicode files):

```
SET TEMPORARY OPTION DEFAULT_ISQL_ENCODING = 'UTF-16'
```

See also

- *OUTPUT Statement [Interactive SQL]* on page 371
- *READ Statement [Interactive SQL]* on page 382

DEFAULT_KB_PER_STRIPE Option

Sets an upper threshold in KB on the amount to write to a stripe before write operations move on to the next stripe.

This setting is the default size for all dbspaces in the IQ main store.

Allowed Values

1 to maximum integer

Default

1

Scope

Option can be set at the database (PUBLIC) level only.

Requires the `SET ANY SYSTEM OPTION` system privilege to set this option. Takes effect immediately.

Description

The default value of 1KB means that one page is compressed and that the compressed page is written to disk as a single operation. Whatever the chosen page size, the next operation writes to the next dbfile in that dbspace.

To write multiple pages to the same stripe before moving to the next stripe, change the `DEFAULT_KB_PER_STRIPE` setting. For example, if the page size is 128KB, and `DEFAULT_KB_PER_STRIPE` set to 512KB, SAP Sybase IQ queues up page writes and writes to disk after reaching the minimum of 512KB of compressed pages.

This option is used only by **CREATE DBSPACE** and defines the default disk striping size for dbspaces in the IQ main store, if **CREATE DBSPACE** does not specify a stripe size.

See also

- *CREATE DBSPACE Statement* on page 117

DEFAULT_LIKE_MATCH_SELECTIVITY_PPM Option

Provides default selectivity estimates (in parts per million) to the optimizer for most **LIKE** predicates.

Allowed Values

0 to 1000000

Default

150000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

DEFAULT_LIKE_MATCH_SELECTIVITY_PPM sets the default selectivity for generic **LIKE** predicates, for example, **LIKE** 'string%string' where % is a wildcard character.

The optimizer relies on this option when other selectivity information is not available and the match string does not start with a set of constant characters followed by a single wildcard.

If the column has either an LF index or a 1- or 2- or 3-byte FP index, the optimizer can get exact information and does not need to use this value.

Users can also specify selectivity in the query. User-supplied condition hints are described in *Reference: Building Blocks, Tables, and Procedures*.

See also

- *DEFAULT_LIKE_RANGE_SELECTIVITY_PPM Option* on page 523
- *FP_LOOKUP_SIZE Option* on page 535

DEFAULT_LIKE_RANGE_SELECTIVITY_PPM Option

Provides default selectivity estimates (in parts per million) to the optimizer for leading constant **LIKE** predicates.

Allowed Values

1 to 1000000

Default

150000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

DEFAULT_LIKE_RANGE_SELECTIVITY_PPM sets the default selectivity for **LIKE** predicates, of the form **LIKE** 'string%' where the match string is a set of constant characters followed by a single wildcard character (%). The optimizer relies on this option when other selectivity information is not available.

If the column has either an LF index or a 1- or 2- or 3-byte FP index, the optimizer can get exact information and does not need to use this value.

Users can also specify selectivity in the query. User-supplied condition hints are described in *Reference: Building Blocks, Tables, and Procedures*.

See also

- *DEFAULT_LIKE_MATCH_SELECTIVITY_PPM Option* on page 522
- *FP_LOOKUP_SIZE Option* on page 535

DEFAULT_PROXY_TABLE_ROW_COUNT Option

Enables you to override the default estimate of the number of rows to return from a proxy table.

Allowed Values

0 to 4294967295

Default

200000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

DEFAULT_TABLE_UDF_ROW_COUNT Option

Enables you to override the default estimate of the number of rows to return from a table UDF (either a C, C++, or Java table UDF).

Allowed Values

0 to 4294967295

Default

200000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

A table UDF can use the **DEFAULT_TABLE_UDF_ROW_COUNT** option to give the query processor an estimate for the number of rows that a table UDF will return. This is the only way a Java table UDF can convey this information. However, for a C or C++ table UDF, the UDF developer should consider publishing this information in the `describe` phase using the `EXTFNAPIV4_DESCRIBE_PARM_TABLE_NUM_ROWS` describe parameter to publish the number of rows it expects to return. The value of `EXTFNAPIV4_DESCRIBE_PARM_TABLE_NUM_ROWS` always overrides the value of the **DEFAULT_PROXY_TABLE_UDF_ROW_COUNT** option.

DELAYED_COMMIT_TIMEOUT Option

Determines when the server returns control to an application following a **COMMIT**.

Allowed Values

Integer, in milliseconds.

Default

500

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option is ignored by SAP Sybase IQ, since DELAYED_COMMITS can only be set OFF.

DELAYED_COMMITS Option

Determines when the server returns control to an application following a **COMMIT**.

Allowed Values

OFF

Default

OFF. This corresponds to ISO COMMIT behavior.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When set to OFF (the only value allowed by SAP Sybase IQ), the application must wait until the **COMMIT** is written to disk. This option must be set to OFF for ANSI/ISO COMMIT behavior.

DISABLE_RI_CHECK Option

Allows load, insert, update, or delete operations to bypass the referential integrity check, improving performance.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporarily for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Users are responsible for ensuring that no referential integrity violation occurs during requests while `DISABLE_RI_CHECK` is set to ON.

DIVIDE_BY_ZERO_ERROR Option [TSQL]

Controls the reporting of division by zero.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option indicates whether division by zero is reported as an error. If the option is set ON, division by zero results in an error with SQLSTATE 22012.

If the option is set OFF, division by zero is not an error; a NULL is returned.

DQP_ENABLED Option

Temporary database option **DQP_ENABLED** allows you to enable or disable distributed query processing at the connection level.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

You can set the temporary database option **DQP_ENABLED** to OFF to disable DQP for the current connection. You can set the option to ON (the default value) to enable DQP for the current connection, but only when DQP is enabled for the user by that user's login policy for the logical server of the current connection.

Setting **DQP_ENABLED** to ON results in an error if DQP is disabled based upon the user's login policy:

```
Invalid setting for option 'DQP_ENABLED'
```

Note: Any changes you make to a user's login policy options affect new connections only. Login policy option settings for existing connections are based upon the time the connection was initially established.

DQP_ENABLED_OVER_NETWORK Option

Temporary database option **DQP_ENABLED_OVER_NETWORK** allows you to enable or disable distributed query processing over the network at the connection level.

Allowed Values

ON, OFF

Default

OFF

Scope

Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role.

Can be set temporary for an individual or public.

Description

You can set the temporary database option **DQP_ENABLED_OVER_NETWORK** to ON to enable DQP over the network for the current connection. The OFF (default) setting has no effect, and the setting of the DQP_ENABLED logical server policy option determines whether or not DQP is used over the network for queries on the current connection.

LS Policy Option Setting	Database Option Setting	DQP Query Behavior
DQP_ENABLED 1	DQP_ENABLED_OVER_NETWORK ON	Queries on the current connection execute over network. Other queries use the shared temporary store.
DQP_ENABLED 2	DQP_ENABLED_OVER_NETWORK ON	All queries execute over network
DQP_ENABLED 2	DQP_ENABLED OFF	All queries run in simplex mode.

Note: Any changes you make to a logical server policy option affect new connections only. Logical server policy options for existing connections are based on the time that the connection was initially established.

EARLY_PREDICATE_EXECUTION Option

Controls whether simple local predicates are executed before query optimization.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If this option is ON (the default), the optimizer finds, prepares, and executes predicates containing only local columns and constraints before query optimization, including join ordering, join algorithm selection, and grouping algorithm selection, so that the values of “Estimated Result Rows” in the query plan are more precise. If this option is OFF, the optimizer finds and prepares the simple predicates, but does not execute them before query optimization. The resulting values of “Estimated Result Rows” are less precise, if the predicates are not executed.

In general, EARLY_PREDICATE_EXECUTION should always be left ON, as this results in improved query plans for many queries.

Note that when EARLY_PREDICATE_EXECUTION is ON, SAP Sybase IQ executes the local predicates for all queries before generating a query plan, even when the NOEXEC option is ON. The generated query plan is the same as the runtime plan.

This information is included in the query plan for the root node:

- Threads used for executing local invariant predicates: if greater than 1, indicates parallel execution of local invariant predicates
- Early_Predicate_Execution: indicates if the option is OFF
- Time of Cursor Creation: the time of cursor creation

The simple predicates whose execution is controlled by this option are referred to as invariant predicates in the query plan. This information is included in the query plan for a leaf node, if there are any local invariant predicates on the node:

- Generated Post Invariant Predicate Rows: actual result after executing local invariant predicate
- Estimated Post Invariant Predicate Rows: calculated by using estimated local invariant predicates selectivity
- Time of Condition Start: starting time of the execution of local invariant predicates

Database Options

- Time of Condition Done: ending time of the execution of local invariant predicates
- Elapsed Condition Time: elapsed time for executing local invariant predicates

ENABLE_ASYNC_IO Option

Allows a DBA to enable or disable the asynchronous IO used by the RLV persistence log.

Allowed Values

TRUE, FALSE

A change in value requires a database close and re-open, or a server restart.

Default

TRUE

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. If permitted, can be set for an arbitrary other user or role, or for all users via the role. Takes effect immediately.

ENABLE_LOB_VARIABLES Option

Controls the data type conversion of large object variables.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

ENABLE_LOB_VARIABLES controls the data type conversion of large object variables.

When ENABLE_LOB_VARIABLES is OFF, large object variables less than 32K are implicitly converted; an error is reported if a large object variable is greater than or equal to 32K. A LONG VARCHAR variable is implicitly converted to a VARCHAR data type and truncated at 32K. A LONG BINARY variable is implicitly converted to a VARBINARY data type and truncated at 32K.

When ENABLE_LOB_VARIABLES is ON, large object variables of any size retain their original data type and size.

Example

Retain the data type and size of large object variables greater than 32K:

```
SET TEMPORARY OPTION ENABLE_LOB_VARIABLES = ON
```

EXTENDED_JOIN_SYNTAX Option

Controls whether queries with an ambiguous syntax for multi-table joins are allowed or are reported as an error.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option reports a syntax error for those queries containing outer joins that have ambiguous syntax due to the presence of duplicate correlation names on a null-supplying table.

This join clause illustrates the kind of query that is reported where C1 is a condition:

```
( R left outer join T , T join S on ( C1 ) )
```

Database Options

If `EXTENDED_JOIN_SYNTAX` is set to `ON`, this query is interpreted as follows, where `C1` and `C2` are conditions:

```
( R left outer join T on ( C1 ) ) join S on ( C2 )
```

FLOATING_POINT_ACCUMULATOR Option

Controls which accumulator to use for **SUM** or **AVG** of floating-point numbers.

Allowed Values

1, 2, 3

Default

2

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY PUBLIC OPTION` system privilege to set this option. Can be set temporarily for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

Setting 1 (fast accumulator) is faster and uses less space for floats and doubles than setting 2. This setting uses a single double precision variable to add double and float numbers, and is subject to the known accuracy limitations of such an approach.

Setting 2 (default) (medium accumulator) uses multiple double precision variables to accumulate floats and doubles. It is very accurate for addends in the range of magnitudes $1e-20$ to $1e20$. While it loses some accuracy outside of this range, it is still accurate enough for most applications. Setting 2 allows the optimizer to choose hash for faster performance more easily than setting 3.

Setting 3 (large accumulator) is highly accurate for all floats and doubles, but its size often precludes the use of hash optimization, which will be a performance limitation for most applications.

FORCE_DROP Option

Causes SAP Sybase IQ to leak, rather than reclaim, database disk space during a **DROP** command.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set as a temporary option only, for an individual connection or for the PUBLIC role, for the duration of the current connection.

Requires SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

You must drop a corrupt index, column or table and set the FORCE_DROP option to ON. This prevents the free list from being incorrectly updated from incorrect or suspect file space allocation information in the object being dropped. After dropping corrupt objects, you can reclaim the file space using the **-iqfrec** and **-iqdropiks** server switches.

When force dropping objects, you must ensure that only the DBA is connected to the database. The server must be restarted immediately after a force drop.

Do not attempt to force drop objects unless SAP Sybase Technical Support has instructed you to do so.

FORCE_DROP procedures for system recovery and database repair are described in *Administration: Backup, Restore, and Data Recovery*.

FORCE_NO_SCROLL_CURSORS Option

Forces all cursors to be non-scrolling.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

By default, all cursors are scrolling. Scrolling cursors with no host variable declared cause SAP Sybase IQ to create a buffer for temporary storage of results. Each row in the result set is stored to allow for backward scrolling.

Setting **FORCE_NO_SCROLL_CURSORS** to ON reduces temporary storage requirements. This option can be useful if you are retrieving very large numbers (millions) of rows. However if your front-end application makes frequent use of backward-scrolling cursor operations, query response will be faster with this option set to OFF.

If your front-end application rarely performs backward-scrolling, make `FORCE_NO_SCROLL_CURSORS = 'ON'` a permanent PUBLIC option, to use less memory and improve query performance.

FORCE_UPDATABLE_CURSORS Option

Controls whether cursors that have not been declared as updatable can be updated.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When **FORCE_UPDATABLE_CURSORS** is ON, cursors which have not been declared as updatable can be updated. This option allows updatable cursors to be used in front-end applications without specifying the **FOR UPDATE** clause of the **DECLARE CURSOR** statement.

Sybase does not recommend the use of **FORCE_UPDATABLE_CURSORS** unless absolutely necessary.

FP_LOOKUP_SIZE Option

Specifies the number of lookup pages and cache memory allocated for Lookup **FP** indexes in SAP Sybase IQ 15. databases.

Allowed Values

1 MB – 4096 MB

Default

16 MB

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Dependencies

FP_LOOKUP_SIZE applies to databases running where

FP_NBIT_IQ15_COMPATIBILITY='ON'. If

FP_NBIT_IQ15_COMPATIBILITY='OFF', the database engine ignores this option.

Description

If FP_NBIT_IQ15_COMPATIBILITY='ON', FP_LOOKUP_SIZE controls the maximum number of lookup pages.

FP_LOOKUP_SIZE must be set public, so the allowed syntax is:

```
SET OPTION public.FP_LOOKUP_SIZE = 1
```

Important: The FP_NBIT_IQ15_COMPATIBILITY option provides tokenized FP support similar to that available in SAP Sybase IQ 15, not complete database compatibility. All SAP Sybase IQ 15 runtime behavior is available using the 16.0 interface. Avoid running a 16.0 database with FP_NBIT_IQ15_COMPATIBILITY='ON'.

See also

- *FP_LOOKUP_SIZE_PPM Option* on page 535
- *MINIMIZE_STORAGE Option* on page 585
- *FP_NBIT_IQ15_COMPATIBILITY Option* on page 539

FP_LOOKUP_SIZE_PPM Option

Controls the amount of main cache allocated to **FP** indexes in SAP Sybase IQ 15. databases.

Allowed Values

1 to 1000000

Default

2500

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Dependencies

FP_LOOKUP_SIZE_PPM applies to databases running where

FP_NBIT_IQ15_COMPATIBILITY='ON'. If

FP_NBIT_IQ15_COMPATIBILITY='OFF', SAP Sybase IQ ignores this option.

Description

If FP_NBIT_IQ15_COMPATIBILITY='ON', FP_LOOKUP_SIZE_PPM controls the maximum number of lookup pages and restricts this number to a parts-per-million value of main memory, that is, the value of FP_LOOKUP_SIZE_PPM * size of main memory / 1,000,000, where the size of main memory is specified by the **-iqmc** server startup parameter.

Important: The FP_NBIT_IQ15_COMPATIBILITY option provides tokenized FP support similar to that available in SAP Sybase IQ 15, not complete database compatibility. All SAP Sybase IQ 15 runtime behavior is available using the 16.0 interface. Avoid running a 16.0 database with FP_NBIT_IQ15_COMPATIBILITY='ON'.

See also

- *FP_LOOKUP_SIZE Option* on page 535
- *MINIMIZE_STORAGE Option* on page 585
- *FP_NBIT_IQ15_COMPATIBILITY Option* on page 539

FP_NBIT_AUTOSIZE_LIMIT Option

Limits the number of distinct values in columns that implicitly load as NBit FP.

Allowed Values

0 – 2,147,483,647

Default

1,048,576

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Dependencies

FP_NBIT_AUTOSIZE_LIMIT applies to databases running where

FP_NBIT_IQ15_COMPATIBILITY='OFF'. If

FP_NBIT_IQ15_COMPATIBILITY='ON', the database engine ignores this option.

Description

FP_NBIT_AUTOSIZE_LIMIT limits the number of distinct values in all newly created columns without an explicit IQ UNIQUE setting. Columns constrained by the FP_NBIT_AUTOSIZE_LIMIT option load with a Flat FP or NBit FP index:

- If FP_NBIT_AUTOSIZE_LIMIT is greater than 0 and less than 2,147,483,647, columns load with an NBit FP index
- If FP_NBIT_AUTOSIZE_LIMIT equals 0, columns load with a Flat FP index

FP_NBIT_AUTOSIZE_LIMIT and FP_NBIT_LOOKUP_MB establish a ceiling for sizing NBit columns during data loads. As long as the number of distinct values is less than FP_NBIT_AUTOSIZE_LIMIT and the total dictionary size (values and counts) is less the FP_NBIT_LOOKUP_MB, the column loads as an NBit. If the load exceeds the FP_NBIT_AUTOSIZE_LIMIT but is less than FP_NBIT_ROLLOVER_MAX_MB, the column rolls over to Flat FP.

DML operations that exceed the FP_NBIT_ROLLOVER_MAX_MB and FP_ENFORCE_LIMITS='ON' rollback and report an error. If the FP_NBIT_ENFORCE_LIMITS='OFF', the column transitions to the next NBit level.

sp_iqindexmetadata returns details about Flat FP or NBit FP columns. **sp_iqrebuildindex** can change explicit or implicit NBit FP column limits, or reformat the default column index as Flat FP or NBit FP.

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqrebuildindex*
- *Reference: Building Blocks, Tables, and Procedures > System Procedures » Alphabetical List of System Stored Procedures > sp_iqindexmetadata*

See also

- *FP_NBIT_ENFORCE_LIMITS Option on page 538*
- *FP_NBIT_IQ15_COMPATIBILITY Option on page 539*

- *FP_NBIT_LOOKUP_MB Option* on page 540
- *FP_NBIT_ROLLOVER_MAX_MB Option* on page 542

FP_NBIT_ENFORCE_LIMITS Option

Enforces sizing limits for explicit and implicit **NBit** columns.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporarily for an individual connection or for the PUBLIC role. Takes effect immediately.

Dependencies

FP_NBIT_ENFORCE_LIMITS applies to databases running where

FP_NBIT_IQ15_COMPATIBILITY='OFF'. If

FP_NBIT_IQ15_COMPATIBILITY='ON', the database engine ignores this option.

Description

DML operations check the FP_NBIT_ENFORCE_LIMITS option when the number of distinct values in a column exceeds the explicit limit set in an IQ UNIQUE constraint, which is above the FP_NBIT_AUTOSIZE value, or when the dictionary size for an implicit **NBit** rollover exceeds the FP_NBIT_ROLLOVER_MAX_MB limit.

- If FP_NBIT_ENFORCE_LIMITS='ON', the DML operation throws an error and rolls back
- If FP_NBIT_ENFORCE_LIMITS='OFF', the DML operation continues and the **NBit** dictionary limits are ignored

sp_iqindexmetadata returns details about Flat FP or NBit FP columns. **sp_iqrebuildindex** can change explicit or implicit NBit FP column limits, or reformat the default column index as Flat FP or NBit FP.

Using **sp_iqrebuildindex** to increase the number of distinct values beyond current limits for a Flat FP column when FP_NBIT_ENFORCE_LIMITS='ON', returns an error. If FP_NBIT_ENFORCE_LIMITS='OFF', **sp_iqrebuildindex** rebuilds the index to the maximum token, which is the largest distinct value.

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqrebuildindex*
- *Reference: Building Blocks, Tables, and Procedures > System Procedures » Alphabetical List of System Stored Procedures > sp_iqindexmetadata*

See also

- *FP_NBIT_AUTOSIZE_LIMIT Option* on page 536
- *FP_NBIT_IQ15_COMPATIBILITY Option* on page 539
- *FP_NBIT_LOOKUP_MB Option* on page 540
- *FP_NBIT_ROLLOVER_MAX_MB Option* on page 542

FP_NBIT_IQ15_COMPATIBILITY Option

Provides support for tokenized **FP** indexes similar to that available in SAP Sybase IQ 15.

Allowed Values

ON, OFF

Default

FP_NBIT_IQ15_COMPATIBILITY='OFF' in all new 16.0 databases.

FP_NBIT_IQ15_COMPATIBILITY='ON' in upgraded SAP Sybase IQ 15 databases.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The FP_NBIT_IQ15_COMPATIBILITY option provides tokenized FP support similar to that available in SAP Sybase IQ 15. All newly created and modified tokenized FP indexes in 16.0 will be NBit. The only 15 style FP (1), FP (2), and FP (3) byte FP indexes available in 16.0 are those from an upgraded database that have had only Read-Only activity.

The FP_NBIT_IQ15_COMPATIBILITY ON/OFF setting only pertains to tokenized FP creation and cut-off behavior:

If FP_NBIT_IQ15_COMPATIBILITY='ON', the database engine:

Database Options

- Enables the `MINIMIZE_STORAGE`, `FP_LOOKUP_SIZE`, `FP_LOOKUP_SIZE_PPM` options
- Creates `DATE` datatypes as `NBit FP`, even if `IQ UNIQUE (0)` is specified
- Rolls over to `Flat FP` at the 3 byte `FP` cutoff (16,777,216 distinct)
- Can tokenize data widths ≤ 255

If `FP_NBIT_IQ15_COMPATIBILITY='OFF'`:

- `MINIMIZE_STORAGE`, `FP_LOOKUP_SIZE`, `FP_LOOKUP_SIZE_PPM` options are ignored.
- `DATE` datatypes are not automatically `NBit`
- Data widths ≤ 32767 may be tokenized
- `NBit FP` (tokenized) upper bound limit is `NBit 31` (2,147,483,648 distinct values)
- `NBit` sizing options determine rollover behavior:
 - `IQ UNIQUE (0)` loads a column as `Flat FP`
 - Columns without `IQ UNIQUE` load as `NBit` up to the auto-size limits
 - Columns where `IQ UNIQUE (n)` is less than the auto-size limit load as `NBit`

Important: The `FP_NBIT_IQ15_COMPATIBILITY` option provides tokenized `FP` support similar to that available in SAP Sybase IQ 15, not complete database compatibility. All SAP Sybase IQ 15 runtime behavior is available using the 16.0 interface. Avoid running a 16.0 database with `FP_NBIT_IQ15_COMPATIBILITY='ON'`.

See also

- *`FP_NBIT_AUTOSIZE_LIMIT` Option* on page 536
- *`FP_NBIT_ENFORCE_LIMITS` Option* on page 538
- *`FP_NBIT_LOOKUP_MB` Option* on page 540
- *`FP_NBIT_ROLLOVER_MAX_MB` Option* on page 542
- *`FP_LOOKUP_SIZE` Option* on page 535
- *`FP_LOOKUP_SIZE_PPM` Option* on page 535
- *`MINIMIZE_STORAGE` Option* on page 585

FP_NBIT_LOOKUP_MB Option

Limits the total dictionary size for implicit `NBit FP` columns.

Allowed Values

1 – 4,294,967,295

Default

64MB

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Dependencies

FP_NBIT_LOOKUP_MB applies to databases running where

FP_NBIT_IQ15_COMPATIBILITY='OFF'. If

FP_NBIT_IQ15_COMPATIBILITY='ON', the database engine ignores this option.

Description

FP_NBIT_AUTOSIZE_LIMIT and FP_NBIT_LOOKUP_MB establish a ceiling for sizing implicit NBit columns. As long as the number of distinct values is less than

FP_NBIT_AUTOSIZE_LIMIT and the total dictionary size (values and counts) is less the FP_NBIT_LOOKUP_MB, the column loads with an NBit FP index. Limits are enforced by the FP_NBIT_ENFORCE_LIMITS option.

DML operations that exceed the FP_NBIT_LOOKUP_MB limit rollover to a Flat FP index.

If an operation exceeds FP_NBIT_LOOKUP_MB and FP_NBIT_ROLLOVER_MAX_MB limits and FP_NBIT_ENFORCE_LIMITS='OFF', the NBit FP transitions to the next NBit level.

sp_iqindexmetadata returns details about Flat FP or NBit FP columns. **sp_iqrebuildindex** can change explicit or implicit NBit FP column limits, or reformat the default column index as Flat FP or NBit FP.

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqrebuildindex*
- *Reference: Building Blocks, Tables, and Procedures > System Procedures » Alphabetical List of System Stored Procedures > sp_iqindexmetadata*

See also

- *FP_NBIT_AUTOSIZE_LIMIT Option* on page 536
- *FP_NBIT_ENFORCE_LIMITS Option* on page 538
- *FP_NBIT_IQ15_COMPATIBILITY Option* on page 539
- *FP_NBIT_ROLLOVER_MAX_MB Option* on page 542

FP_NBIT_ROLLOVER_MAX_MB Option

Sets a threshold for the total dictionary size for implicit NBit rollovers to Flat FP.

Allowed Values

1 – 4,294,967,295

Default

16384

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Dependencies

FP_NBIT_ROLLOVER_MAX_MB applies to databases running where

FP_NBIT_IQ15_COMPATIBILITY='OFF'. If

FP_NBIT_IQ15_COMPATIBILITY='ON', the database engine ignores this option.

Description

FP_NBIT_AUTOSIZE_LIMIT and FP_NBIT_LOOKUP_MB establish a ceiling for sizing implicit NBit FP columns. DML operations that exceed these values check the FP_NBIT_ROLLOVER_MAX_MB limit, which sets the dictionary size (values and counts) for implicit NBit rollovers:

- If the total dictionary size does not exceed the FP_NBIT_ROLLOVER_MAX_MB, the NBit column rolls over to a Flat FP.
- If the dictionary size exceeds the FP_NBIT_ROLLOVER_MAX_MB limit and FP_NBIT_ENFORCE_LIMITS='ON', DML operations throw an error and roll back.
- If the dictionary size exceeds the FP_NBIT_ROLLOVER_MAX_MB limit and FP_NBIT_ENFORCE_LIMITS='OFF' (default), DML operations throw an error, and the NBit dictionary continues to grow.
- If FP_NBIT_ROLLOVER_MAX_MB='0', the NBit column rolls over to Flat FP.

sp_iqindexmetadata returns details about Flat FP or NBit FP columns. **sp_iqrebuildindex** can change explicit or implicit NBit FP column limits, or reformat the default column index as Flat FP or NBit FP.

Additional Information

- *Reference: Building Blocks, Tables, and Procedures > System Procedures > Alphabetical List of System Stored Procedures > sp_iqrebuildindex*
- *Reference: Building Blocks, Tables, and Procedures > System Procedures » Alphabetical List of System Stored Procedures > sp_iqindexmetadata*

FP_PREDICATE_WORKUNIT_PAGES Option

Specifies degree of parallelism used in the default index.

Allowed Values

Integer

Default

200

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The default index calculates some predicates such as SUM, RANGE, MIN, MAX and COUNT DISTINCT in parallel. FP_PREDICATE_WORKUNIT_PAGES affects the degree of parallelism used by specifying the number of pages worked on by each thread. To increase the degree of parallelism, decrease the value of this option.

FPL_EXPRESSION_MEMORY_KB Option

Controls the use of memory for the optimization of queries involving functional expressions against columns having enumerated storage.

Allowed Values

0 – 20000

Default

1024 kilobytes

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

FPL_EXPRESSION_MEMORY_KB controls the use of memory for the optimization of queries involving functional expressions against columns having enumerated storage. The option enables the DBA to constrain the memory used by this optimization and balance it with other SAP Sybase IQ memory requirements, such as caches. Setting this option to 0 switches off optimization.

GARRAY_FILL_FACTOR_PERCENT Option

Specifies the percent of space on each **HG** garray page to reserve for future incremental inserts into existing groups.

Allowed Values

0 – 1000

Default

25

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The garray tries to pad out each group to include a pad of empty space set by the value. This space is used for rows added to existing index groups.

An **HG** index can reserve some storage on a per-group basis (where group is defined as a group of rows with equivalent values). Reserving space consumes additional disk space, but can help the performance of incremental inserts into the **HG** index.

If you plan to do future incremental inserts into an **HG** index, and those new rows have values that are already present in the index, a nonzero value for this option might improve incremental insert performance.

If you do not plan to incrementally update the index, you can reduce the values of this option to save disk space.

See also

- *GARRAY_PAGE_SPLIT_PAD_PERCENT* Option on page 545

GARRAY_INSERT_PREFETCH_SIZE Option

Specifies number of pages used for prefetch.

Allowed Values

0 – 100

Default

3

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option defines the number of database pages read ahead during an insert to a column that has an **HG** index.

Do not set this option unless advised to do so by Sybase Technical Support.

See also

- *GARRAY_FILL_FACTOR_PERCENT* Option on page 544

GARRAY_PAGE_SPLIT_PAD_PERCENT Option

Determines per-page fill factor during page splits on the garray and specifies the percent of space on each **HG** garray page to reserve for future incremental inserts.

Allowed Values

0 – 100

Default

25

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Splits of a garray page try to leave that percentage empty. This space is used for rows added to new index groups.

An **HG** index can reserve storage at the page level that can be allocated to new groups when additional rows are inserted. Reserving space consumes additional disk space, but can help the performance of incremental inserts into the **HG** index.

If future plans include incremental inserts into an **HG** index, and the new rows do not have values that are already present in the index, a nonzero value for `GARRAY_PAGE_SPLIT_PAD_PERCENT` could improve incremental insert performance.

If you do not plan to incrementally update the index, you can reduce the values of this option to save disk space.

See also

- *GARRAY_FILL_FACTOR_PERCENT* Option on page 544

GARRAY_RO_PREFETCH_SIZE Option

Specifies number of pages used for prefetch.

Allowed Values

0 – 100

Default

10

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option defines the number of database pages read ahead during a query to a column that has an **HG** index.

Do not set this option unless advised to do so by Sybase Technical Support.

HASH_PINNABLE_CACHE_PERCENT Option

Controls the maximum percentage of a user's temp memory that a hash object can pin.

Allowed Values

0 – 100

Default

20

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

HASH_PINNABLE_CACHE_PERCENT controls the percentage of a user's temp memory allocation that any one hash object can pin in memory. The default is 20%, but you should reduce this number to 10% if you are running complex queries, or increase this number to 50% if you have simple queries that need a single large hash object to run, such as a large **IN** subquery.

HASH_PINNABLE_CACHE_PERCENT is for use by primarily Sybase Technical Support. If you change the value of it, do so with extreme caution; first analyze the effect on a wide variety of queries.

See also

- *BIT_VECTOR_PINNABLE_CACHE_PERCENT Option* on page 489
- *SORT_PINNABLE_CACHE_PERCENT Option* on page 629

HASH_THRASHING_PERCENT Option

Specifies the percent of hard disk I/Os allowed during the execution of a statement that includes a query involving hash algorithms, before the statement is rolled back and an error message is reported.

Allowed Values

0 – 100

Default

10

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If a query that uses hash algorithms causes an excessive number of hard disk I/Os (paging buffers from memory to disk), query performance is negatively affected, and server performance might also be affected. HASH_THRASHING_PERCENT controls the percentage of hard disk I/Os allowed before the statement is rolled back and an error message is returned. The text of the error message is either `Hash insert thrashing detected` or `Hash find thrashing detected`.

The default value of HASH_THRASHING_PERCENT is 10%. Increasing this value permits more paging to disk before a rollback and decreasing this value permits less paging before a rollback.

See also

- *HASH_PINNABLE_CACHE_PERCENT Option on page 547*

HG_DELETE_METHOD Option

Specifies the algorithm used during a delete in a **HG** index.

Allowed Values

0 – 3

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option chooses the algorithm used by the **HG** index during a delete operation. The cost model considers the CPU related costs as well as I/O related costs in selecting the appropriate delete algorithm. The cost model takes into account:

- Rows deleted
- Index size
- Width of index data type
- Cardinality of index data
- Available temporary cache
- Machine related I/O and CPU characteristics
- Available CPUs and threads
- Referential integrity costs

To force a “small” method, set this option to 1. To force the “large” method, set the option to 2. To force a “midsize” method, set the option to 3.

HG_SEARCH_RANGE Option

Specifies the maximum number of Btree pages used in evaluating a range predicate in the **HG** index.

Allowed Values

Integer

Default

10

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Database Options

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporarily for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The default setting of this option is appropriate for most queries.

This option effectively controls the amount of time the optimizer spends searching for the best index to use for a range predicate. Setting this option higher may cause a query to spend more time in the optimizer, but as a result may choose a better index to resolve a range predicate.

HTTP_SESSION_TIMEOUT Option

Specifies the amount of time, in minutes, that the client waits for an HTTP session to time out before giving up.

Allowed Values

Integer (0 – 525600)

Default

30

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Takes effect immediately.

Description

This option provides variable session timeout control for Web service applications. A Web service application can change the timeout value from within any request that owns the HTTP session, but a change to the timeout value can impact subsequent queued requests if the HTTP session times out. The Web application must include logic to detect whether a client is attempting to access an HTTP session that no longer exists. This can be done by examining the value of the **SessionCreateTime** connection property to determine whether a timestamp is valid: if the HTTP request is not associated with the current HTTP session, the **SessionCreateTime** connection property contains an empty string.

IDENTITY_ENFORCE_UNIQUENESS Option

Creates a unique **HG** index on each IDENTITY/AUTOINCREMENT column, if the column is not already a primary key.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When option is set ON, **HG** indexes are created on future identity columns. The index can only be deleted if the deleting user is the only one using the table and the table is not a local temporary table.

See also

- *QUERY_PLAN Option* on page 608

IDENTITY_INSERT Option

Enables users to insert values into or to update an IDENTITY or AUTOINCREMENT column.

Allowed Values

= 'tablename'

Default

Option not set.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

Database Options

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Note: If you set a user level option for the current option, the corresponding temporary option is also set. See *Scope and Duration of Database Options*.

Description

When IDENTITY_INSERT is set, insert/update is enabled. A table name must be specified to identify the column to insert or update. If you are not the table owner, qualify the table name with the owner name.

To drop a table with an IDENTITY column, IDENTITY_INSERT must not be set to that table.

Examples

If you use the table Employees to run explicit inserts:

```
SET TEMPORARY OPTION IDENTITY_INSERT = 'DBA.Employees'
```

To turn the option off, specify the equals sign and an empty string:

```
SET TEMPORARY OPTION IDENTITY_INSERT = ''
```

Illustrates the effect of user level options on temporary options (see *Note*), if you are connected to the database as DBA and enter:

```
SET OPTION IDENTITY_INSERT = 'Customers'
```

The value for the option is set to Customers for the user DBA and temporary for the current connection. Other users who subsequently connect to the database as DBA find their option value for IDENTITY_INSERT is Customers also.

See also

- *Scope and Duration of Database Options* on page 465
- *QUERY_PLAN Option* on page 608

IN_SUBQUERY_PREFERENCE Option

Controls the choice of algorithms for processing an **IN** subquery.

Allowed Values

-3 to 3

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The IQ optimizer has a choice of several algorithms for processing **IN** subqueries. This option allows you to override the optimizer's costing decision when choosing the algorithm to use. It does not override internal rules that determine whether an algorithm is legal within the query engine.

IN_SUBQUERY_PREFERENCE is normally used for internal testing and for manually tuning queries that the optimizer does not handle well. Only experienced DBAs should use it. The only reason to use this option is if the optimizer seriously underestimates the number of rows produced by a subquery, and the hash object is thrashing. Before setting this option, try to improve the mistaken estimate by looking for missing indexes and dependent predicates.

Inform Sybase Technical Support if you need to set IN_SUBQUERY_PREFERENCE, as setting this option might mean that a change to the optimizer is appropriate.

Table 16. IN_SUBQUERY_PREFERENCE Valid Values

Value	Action
0	Let the optimizer choose
1	Prefer sort-based IN subquery
2	Prefer vertical IN subquery (where a subquery is a child of a leaf node in the query plan)
3	Prefer hash-based IN subquery
-1	Avoid sort-based IN subquery
-2	Avoid vertical IN subquery
-3	Avoid hash-based IN subquery

INDEX_ADVISOR Option

Generates messages suggesting additional column indexes that may improve performance of one or more queries.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When set ON, the index advisor prints index recommendations as part of the query plan or as a separate message in the message log file, if query plans are not enabled. These messages begin with the string “Index Advisor:” and you can use that string to search and filter them from a message file. The output is in OWNER.TABLE.COLUMN format.

Set both **INDEX_ADVISOR** and **INDEX_ADVISOR_MAX_ROWS** to accumulate index advice.

Note: When **INDEX_ADVISOR_MAX_ROWS** is set ON, index advice will not be written to the message file as separate messages. Advice will, however, continue to be displayed on query plans in the message file.

Table 17. Index Advisor

Situation	Recommendation
Local predicates on a single column where an HG , LF , HNG , DATE , TIME or DATETIME index would be desirable, as appropriate.	Recommend adding an <index-type> index to column <col>
Single column join keys where an LF or HG index would be useful.	Add an LF or HG index to join key <col>
Single column candidate key indexes where a HG exists, but could be changed to a unique HG or LF	Change join key <col> to a unique LF or HG index

Situation	Recommendation
Join keys have mismatched data types, and regenerating one column with a matched data type would be beneficial.	Make join keys <col1> and <col2> identical data types
Subquery predicate columns where an LF or HG index would be useful.	Add an LF or HG index to subquery column <col>
Grouping columns where an LF or HG index would be useful.	Create an LF or HG index on grouping column <col>
Single-table intercolumn comparisons where the two columns are identical data types, a CMP index are recommended.	Create a CMP index on <col1>, <col2>
) Columns where an LF or HG index exists, and the number of distinct values allows, suggest converting the FP to a 1 or 2-byte FP index.	Rebuild <col> with 'optimize storage=on'
To support the lookup of default indexes three bytes wide	Rebuild your FP Index as a 3-byte FP with an IQ UNIQUE constraint value of 65537

It is up to you to decide how many queries benefit from the additional index and whether it is worth the expense to create and maintain the indexes. In some cases, you cannot determine how much, if any, performance improvement results from adding the recommended index.

For example, consider columns used as a join key. SAP Sybase IQ uses metadata provided by **HG** or **LF** indexes extensively to generate better/faster query plans to execute the query. Putting an **HG** or **LF** index on a join column without one makes the IQ optimizer far more likely to choose a faster join plan, but without adding the index and running the query again, it is very hard to determine whether query performance stays the same or improves with the new index.

Example

Index advisor output with query plan set OFF:

```
I. 03/30 14:18:45. 0000000002 Advice: Add HG or LF index
on DBA.ta.c1 Predicate: (ta2.c1 < BV(1))
```

Index advisor output with query plan set ON:

Note: This method accumulates index advisor information for multiple queries, so that advice for several queries can be tracked over time in a central location.

```
I. 03/30 14:53:24. 0000000008 [20535]: 6      ...#03: Leaf
I. 03/30 14:53:24. 0000000008 [20535]:      Table Name: tb
I. 03/30 14:53:24. 0000000008 [20535]:      Condition 1
(Invariant):
(tb.c3 =tb.c4)
I. 03/30 14:53:24. 0000000008 [20535]:      Condition 1 Index
Advisor:
Add a CMP index on DBA.tb (c3,c4)
```

See also

- *FP_LOOKUP_SIZE Option* on page 535
- *INDEX_ADVISOR_MAX_ROWS Option* on page 556
- *QUERY_PLAN Option* on page 608

INDEX_ADVISOR_MAX_ROWS Option

Sets the maximum number of unique advice messages stored by the index advisor to `max_rows`.

Allowed Values

Value	Description
0	Minimum value disables collection of index advice
4294967295	Maximum value allowed

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

INDEX_ADVISOR_MAX_ROWS limits the number of messages stored by the index advisor. Once the specified limit has been reached, the **INDEX_ADVISOR** will not store new advice. It will, however, continue to update counts and timestamps for existing advice messages.

```
SET OPTION public.Index_Advisor_Max_Rows = max_rows;
```

See also

- *FP_LOOKUP_SIZE Option* on page 535
- *INDEX_ADVISOR Option* on page 554

INDEX_PREFERENCE Option

Controls the choice of indexes to use for queries.

Allowed Values

Value	Action
0	Let the optimizer choose
1	Prefer LF indexes
2	Prefer HG indexes
3	Prefer HNG indexes
4	Prefer CMP indexes
5	Prefer the default index
6	Prefer WD indexes
8	Prefer DATE indexes
9	Prefer TIME indexes
10	Prefer DTTM indexes
-1	Avoid LF indexes
-2	Avoid HG indexes
-3	Avoid HNG indexes
-4	Avoid CMP indexes
-5	Avoid the default index
-6	Avoid WD indexes
-8	Avoid DATE indexes
-9	Avoid TIME indexes
-10	Avoid DTTM indexes

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

Database Options

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The SAP Sybase IQ optimizer normally chooses the best index available to process local **WHERE** clause predicates and other operations that can be done within an IQ index.

INDEX_PREFERENCE is used to override the optimizer choice for testing purposes; under most circumstances, it should not be changed.

INFER_SUBQUERY_PREDICATES Option

Controls the optimizer's inference of additional subquery predicates.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

INFER_SUBQUERY_PREDICATES controls whether the optimizer is allowed to infer additional subquery predicates from an existing subquery predicate through transitive closure across a simple equality join predicate. In most cases in which the optimizer chooses to make this inference, the query runs faster. There are some exceptions to this performance improvement, so you may need to experiment to be sure that this option is appropriate for your environment.

IQGOVERN_MAX_PRIORITY Option

Limits the allowed `IQGOVERN_PRIORITY` setting.

Allowed Values

1 – 3

Default

2

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Limits the allowed `IQGOVERN_PRIORITY` setting, which affects the order in which a user's queries are queued for execution. In the range of allowed values, 1 indicates high priority, 2 (the default) medium priority, and 3 low priority. SAP Sybase IQ returns an error if a user sets `IQGOVERN_PRIORITY` higher than `IQGOVERN_MAX_PRIORITY`.

See also

- *IQGOVERN_PRIORITY Option* on page 559
- *IQGOVERN_PRIORITY_TIME Option* on page 560

IQGOVERN_PRIORITY Option

Assigns a priority to each query waiting in the `-iqgovern` queue.

Allowed Values

1 – 3

Default

2

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

Database Options

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Assigns a value that determines the order in which a user's queries are queued for execution. In the range of allowed values, 1 indicates high priority, 2 (the default) medium priority, and 3 low priority. This switch can be set temporary per user or public by any user. Queries with a lower priority will not run until all higher priority queries have executed.

This option is limited by the per user or per group value of the option IQGOVERN_MAX_PRIORITY.

See also

- *IQGOVERN_MAX_PRIORITY Option* on page 559
- *IQGOVERN_PRIORITY_TIME Option* on page 560

IQGOVERN_PRIORITY_TIME Option

Limits the time a high priority query waits in the queue before starting.

Allowed Values

0 – 1,000,000 seconds. Must be lower than IQGOVERN_MAX_PRIORITY.

Default

0 (disabled)

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

Limits the time a high priority (priority 1) query waits in the queue before starting. When the limit is reached, the query is started even if it exceeds the number of queries allowed by the **-iqgovern** setting. The range is from 1 to 1,000,000 seconds. The default (0) disables this feature. IQGOVERN_PRIORITY_TIME must be set PUBLIC.

See also

- *IQGOVERN_MAX_PRIORITY Option* on page 559
- *IQGOVERN_PRIORITY Option* on page 559

ISOLATION_LEVEL Option

Controls the locking isolation level for catalog store tables.

Allowed Values

- 0 – Allow dirty reads, nonrepeatable reads, and phantom rows.
- 1 – Prevent dirty reads. Allow nonrepeatable reads and phantom rows.
- 2 – Prevent dirty reads and guarantee repeatable reads. Allow phantom rows.
- 3 – Serializable. Do not allow dirty reads, guarantee repeatable reads, and do not allow phantom rows.

Default

0

1 for Open Client and JDBC connections

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

ISOLATION_LEVEL determines the isolation level for tables in the catalog store. SAP Sybase IQ always enforces level 3 for tables in the IQ store. Level 3 is equivalent to ANSI level 4.

JAVA_LOCATION Option

Specifies the path of the Java VM for the database.

Allowed Values

String

Default

Empty string

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

By default, this option contains an empty string. In this case, the database server searches the `JAVA_HOME` environment variable, the path, and other locations for the Java VM.

See also

- *JAVA_VM_OPTIONS Option* on page 562

JAVA_VM_OPTIONS Option

Specifies command line options that the database server uses when it launches the Java VM.

Allowed Values

String

Default

Empty string

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

`JAVA_VM_OPTIONS` specifies options that the database server uses when launching the Java VM specified by the `JAVA_LOCATION` option. These additional options can be used to set up the Java VM for debugging purposes or to run as a service on UNIX platforms. In some cases, additional options are required to use the Java VM in 64-bit mode instead of 32-bit mode.

See also

- *JAVA_LOCATION Option* on page 561

JOIN_EXPANSION_FACTOR Option

Controls how conservative the optimizer's join result estimates are in unusually complex situations.

Allowed Values

1 – 100

Default

30

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option controls how conservative the join optimizer's result size estimates are in situations where an input to a specific join has already passed through at least one intermediate join that can result in multiple copies of rows projected from the table being joined.

A level of zero indicates that the optimizer should use the same estimation method above intermediate expanding joins as it would if there were no intermediate expanding joins.

This results in the most aggressive (small) join result size estimates.

A level of 100 indicates that the optimizer should be much more conservative in its estimates whenever there are intermediate expanding joins, and this results in the most conservative (large) join result size estimates.

Normally, you should not need to change this value. If you do, set JOIN_EXPANSION_FACTOR as a temporary or user option.

JOIN_OPTIMIZATION Option

Enables or disables the optimization of the join order.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When JOIN_OPTIMIZATION is ON, SAP Sybase IQ optimizes the join order to reduce the size of intermediate results and sorts, and to balance the system load. When the option is OFF,

Database Options

the join order is determined by the order of the tables in the **FROM** clause of the **SELECT** statement.

`JOIN_OPTIMIZATION` should always be set `ON`.

`JOIN_OPTIMIZATION` controls the order of the joins, but not the order of the tables. To show the distinction, consider this example **FROM** clause with four tables:

```
FROM A, B, C, D
```

By default, this **FROM** clause creates a left deep plan of joins that could also be explicitly represented as:

```
FROM ((A, B), C), D)
```

If `JOIN_OPTIMIZATION` is turned `OFF`, then the order of these joins on the sets of tables is kept precisely as specified in the **FROM** clause. Thus A and B must be joined first, then that result must be joined to table C, and then finally joined to table D. This option does not control the left/right orientation at each join. Even with `JOIN_OPTIMIZATION` turned `OFF`, the optimizer, when given the above **FROM** clause, can produce a join plan that looks like:

```
FROM ((C, (A, B)), D)
```

or

```
FROM ((B, A), C), D)
```

or

```
FROM (D, ((A, B), C))
```

In all of these cases, A and B are joined first, then that result is joined to C, and finally that result is joined to table D. The order of the joins remains the same, but the order of the tables appears different.

In general, if `JOIN_OPTIMIZATION` is turned `OFF`, you probably should use parentheses in the **FROM** clause, as in the above examples, to make sure that you get the join order you want. If you want to join A and B to the join of C and D, you can specify this join by using parentheses:

```
FROM ((A, B), (C, D))
```

Note that the above **FROM** clause is a different join order than the original example **FROM** clause, even though all the tables appear in the same order.

`JOIN_OPTIMIZATION` should be set to `OFF` only to diagnose obscure join performance issues or to manually optimize a small number of predefined queries. With `JOIN_OPTIMIZATION` turned `OFF`, queries can join up to 128 tables, but might also suffer serious performance degradation.

Warning! If you turn off `JOIN_OPTIMIZATION`, SAP Sybase IQ has no way to ensure optimal performance for queries containing joins. You assume full responsibility for performance aspects of your queries.

JOIN_PREFERENCE Option

Controls the choice of algorithms when processing joins.

Allowed Values

Value	Action
0	Let the optimizer choose
1	Prefer sort-merge
2	Prefer nested-loop
3	Prefer nested-loop push-down
4	Prefer hash
5	Prefer hash push-down
6	Prefer asymmetric sort-merge join
7	Prefer sort-merge push-down
8	Prefer asymmetric sort-merge push-down join
9	Prefer partitioned hash join if the join keys include all the partition keys of a hash partitioned table
10	Prefer partitioned hash-push down join if the join keys include all the partition keys of a hash partitioned table
11	Prefer partitioned sort-merge join if the join keys include all the partition keys of a hash partitioned table
12	Prefer partitioned sort-merge push-down join if the join keys include all the partition keys of a hash partitioned table
-1	Avoid sort-merge
-2	Avoid nested-loop
-3	Avoid nested-loop push-down
-4	Avoid hash
-5	Avoid hash push-down
-6	Avoid asymmetric sort-merge join
-7	Avoid sort-merge push-down
-8	Avoid asymmetric sort-merge push-down join

Value	Action
-9	Avoid partitioned hash join if the join keys include all the partition keys of a hash partitioned table
-10	Avoid partitioned hash-push down join if the join keys include all the partition keys of a hash partitioned table
-11	Avoid partitioned sort-merge join if the join keys include all the partition keys of a hash partitioned table
-12	Avoid partitioned sort-merge push-down join if the join keys include all the partition keys of a hash partitioned table

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

For joins within a query, the SAP Sybase IQ optimizer has a choice of several algorithms for processing the join. JOIN_PREFERENCE allows you to override the optimizer's cost-based decision when choosing the algorithm to use. It does not override internal rules that determine whether an algorithm is legal within the query engine. If you set it to any nonzero value, every join in a query is affected; you cannot use it to selectively modify one join out of several in a query, but join condition hint strings can do so.

This option is normally used for internal testing or tuning of report queries, and only experienced DBAs should use it.

Simple equality join predicates can be tagged with a predicate hint that allows a join preference to be specified for just that one join. If the same join has more than one join condition with a local join preference, and if those hints are not the same value, then all local preferences are ignored for that join. Local join preferences do not affect the join order chosen by the optimizer.

This example requests a hash join:

```
AND (T.X = 10 * R.x, 'J:4')
```


JOIN_SIMPLIFICATION_THRESHOLD Option

Controls the minimum number of tables being joined together before any join optimizer simplifications are applied.

Allowed Values

1 – 64

Default

12

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The query optimizer simplifies its optimization of join order by separate handling of both lookup tables (that is, nonselective dimension tables) and tables that are effective Cartesian products. After simplification, it optimizes the remaining tables for join order, up to the limit set by MAX_JOIN_ENUMERATION.

Setting this option to a value greater than the current value for MAX_JOIN_ENUMERATION has no effect.

Setting this value below the value for MAX_JOIN_ENUMERATION might improve the time required to optimize queries containing many joins, but may also prevent the optimizer from finding the best possible join plan.

If you change this value, set the JOIN_SIMPLIFICATION_THRESHOLD as a temporary or user option, and to a value of at least 9.

See also

- *MAX_JOIN_ENUMERATION Option* on page 579

LF_BITMAP_CACHE_KB Option

Specifies the amount of memory to use for a load into a LF index.

Allowed Values

1 – 8

Database Options

Default

4

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

LF_BITMAP_CACHE_KB defines the amount of heap memory (in KB) per distinct value used during a load into an LF index. The default allots 4KB. If the sum of the distinct counts for all LF indexes on a particular table is relatively high (greater than 10,000), then heap memory use might increase to the point of impacting load performance due to system page faulting. If this is the case, reduce the value of LF_BITMAP_CACHE_KB.

This formula shows how to calculate the heap memory used (in bytes) by a particular LF index during a load:

```
Heap-memory-used = (lf_bitmap_cache_kb * 1024)
* lf-distinct-count-for-column
```

Using the default of 4KB, an LF index with 1000 distinct values can use up to 4MB of heap memory during a load.

LOAD_ZEROLENGTH_ASNULL Option

Specifies LOAD statement behavior under certain conditions.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option specifies LOAD statement behavior under these conditions:

- inserting a zero-length data value into a column of data type CHAR, VARCHAR, LONG VARCHAR, BINARY, VARBINARY, or LONG BINARY
and
- a NULL column-spec; for example, NULL(ZEROS) or NULL(BLANKS) is also given for that same column

Set LOAD_ZEROLENGTH_ASNNULL ON to load a zero-length value as NULL when the above conditions are met.

Set LOAD_ZEROLENGTH_ASNNULL OFF to load a zero-length value as zero-length, subject to the setting of option NON_ANSI_NULL_VARCHAR.

See also

- *NON_ANSI_NULL_VARCHAR Option* on page 593
- *LOAD TABLE Statement* on page 343

LOG_CONNECT Option

Controls logging of user connections.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description

When this option is ON, a message appears in the IQ message log (.iqmsg file) every time a user connects to or disconnects from the SAP Sybase IQ database.

Note: If this option is set OFF (connection logging disabled) when a user connects, and then turned on before the user disconnects, the message log shows that user disconnecting but not connecting.

LOG_CURSOR_OPERATIONS Option

Controls logging of cursor operations.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When this option is ON, a message appears in the IQ message log every time you open or close a cursor. Normally this option should be OFF, which is the default. Turn it ON only if you are having a problem and must provide debugging data to Sybase Technical Support.

LOG_DEADLOCKS Option

Controls whether deadlock reporting is turned on or off.

Allowed values

On, Off

Default

Off

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Remarks

When this option is set to On, the database server logs information about deadlocks in an internal buffer. The size of the buffer is fixed at 10000 bytes. You can view the deadlock

information using the `sa_report_deadlocks` stored procedure. The contents of the buffer are retained when this option is set to Off.

When deadlock occurs, information is reported for only those connections involved in the deadlock. The order in which connections are reported is based on which connection is waiting for which row. For thread deadlocks, information is reported about all connections.

When you have deadlock reporting turned on, you can also use the Deadlock system event to take action when a deadlock occurs

LOGIN_MODE Option

Controls the use of integrated logins for the database.

Allowed Values

- Standard – the default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
- Mixed – both integrated logins and standard logins are allowed.
- Integrated – all logins to the database must be made using integrated logins.
- Kerberos – all logins to the database must be made using Kerberos logins.
- LDAPUA – all logins to the database must be made using LDAP logins.

Note: Mixed is equivalent to "Standard,Integrated".

Default

Standard

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description

Values are case-insensitive:

Warning!

- Restricting the `LOGIN_MODE` to a single mode in a mixed environment (for example, Integrated only or LDAPUA only) restricts connections to only those users who have been granted the corresponding login mapping. Attempting to connect using other methods generates an error. The only exceptions to this are users with full administrative rights (`SYS_AUTH_DBA_ROLE` or `SYS_AUTH_SSO_ROLE`).
- Restricting the `LOGIN_MODE` to LDAPUA only may result in a configuration where no users can connect to the server if no user or login policy exists that permits LDAPUA. Use

the command line switch **-aluser-id-list** with the **start_iq** utility to recover from this situation.

LOGIN_PROCEDURE Option

Specifies a login procedure that sets connection compatibility options at start-up.

Allowed Values

String

Default

sp_login_environment system procedure

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The initial connection compatibility options settings are controlled using the LOGIN_PROCEDURE option, which is called after all the checks have been performed to verify that the connection is valid. The LOGIN_PROCEDURE option names a stored procedure to run when users connect. The default setting is to use the **sp_login_environment** system stored procedure. You can specify a different stored procedure. The procedure specified by the LOGIN_PROCEDURE option is not executed for event connections.

The **sp_login_environment** procedure checks to see if the connection is being made over TDS. If the connection is made over TDS, **sp_login_environment** calls the **sp_tsq_environment** procedure, which sets several options to new default values for the current connection.

See also

- *Initial Option Settings* on page 468

MAIN_RESERVED_DBSPACE_MB Option

Controls the amount of space SAP Sybase IQ reserves in the IQ main store.

Allowed Values

Integer greater than or equal to 200, in megabytes

Default

200; SAP Sybase IQ actually reserves a maximum of 50% and a minimum of 1% of the last read-write file in `IQ_SYSTEM_MAIN`

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

`MAIN_RESERVED_DBSPACE_MB` controls the amount of space SAP Sybase IQ sets aside in the IQ main store for certain small but critical data structures used during release savepoint, commit, and checkpoint operations. For a production database, set this value between 200MB and 1GB, or at least 20 percent of `IQ_SYSTEM_MAIN` size. The larger your IQ page size and number of concurrent connections, the more reserved space you need.

Reserved space size is calculated as a maximum of 50 percent and a minimum of 1 percent of the last read-write file in `IQ_SYSTEM_MAIN`.

SAP Sybase IQ ignores the `MAIN_RESERVED_DBSPACE_MB` option if the actual dbspace size is less than twice the size of the `MAIN_RESERVED_DBSPACE_MB` value. In dbspaces less than 100MB (such as the demo database), half the usable space may be reserved.

MAX_CARTESIAN_RESULT Option

Limits the number of rows resulting from a Cartesian join.

Allowed Values

Any integer

Default

100000000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

MAX_CARTESIAN_RESULT limits the number of result rows from a query containing a Cartesian join (usually the result of missing one or more join conditions when creating the query). If SAP Sybase IQ cannot find a query plan for the Cartesian join with an estimated result under this limit, it rejects the query and returns an error. Setting MAX_CARTESIAN_RESULT to 0 disables the check for the number of result rows of a Cartesian join.

MAX_CLIENT_NUMERIC_PRECISION Option

Controls the maximum precision for numeric data sent to the client.

Allowed Values

0 – 126

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When SAP Sybase IQ performs its calculation, it promotes data types to an appropriate size that ensures accuracy. The promoted data type might be larger in size than Open Client and some ODBC applications can handle correctly.

When MAX_CLIENT_NUMERIC_PRECISION is a nonzero value, SAP Sybase IQ checks that numeric result columns do not exceed this value. If the result column is bigger than MAX_CLIENT_NUMERIC_PRECISION allows, and SAP Sybase IQ cannot cast it to the specified precision, the query returns this error:

```
Data Exception - data type conversion is not possible %1  
SQLCODE = -1001006
```

Note: In SQL Anywhere, the maximum value supported for the numeric function is 255. If the precision of the numeric function exceeds the maximum value supported, you see the error: The result datatype for function '_funcname' exceeds the maximum supported numeric precision of 255. Please set the proper value for precision in numeric function, 'location'

See also

- *MAX_CLIENT_NUMERIC_SCALE Option* on page 575
- *PRECISION Option* on page 602

MAX_CLIENT_NUMERIC_SCALE Option

Controls the maximum scale for numeric data sent to the client.

Allowed Values

0 – 126

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When SAP Sybase IQ performs its calculation, it promotes data types to an appropriate scale and size that ensure accuracy. The promoted data type might be larger than the original defined data size. You can set this option to the scale you want for numeric results.

Multiplication, division, addition, subtraction, and aggregate functions can all have results that exceed the maximum precision and scale.

For example, when a DECIMAL(88,2) is multiplied with a DECIMAL(59,2), the result could require a DECIMAL(147,4). With MAX_CLIENT_NUMERIC_PRECISION of 126, only 126 digits are kept in the result. If MAX_CLIENT_NUMERIC_SCALE is 4, the results are returned as a DECIMAL(126,4). If MAX_CLIENT_NUMERIC_SCALE is 2, the result are returned as a DECIMAL(126,2). In both cases, there is a possibility for overflow.

See also

- *MAX_CLIENT_NUMERIC_PRECISION Option* on page 574
- *SCALE Option* on page 626

MAX_CUBE_RESULT Option

Sets the maximum number of rows that the IQ optimizer considers for a **GROUP BY CUBE** operation.

Allowed Values

0 – 4294967295

Default

10000000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When generating a query plan, the IQ optimizer estimates the total number of groups generated by the **GROUP BY CUBE** hash operation. The IQ optimizer uses a hash algorithm for the **GROUP BY CUBE** operation. This option sets an upper boundary for the number of estimated rows the optimizer considers for a hash algorithm that can be run. If the actual number of rows exceeds the MAX_CUBE_RESULT value, the optimizer stops processing the query and returns the error Estimate number: *nnn* exceeds the default MAX_CUBE_RESULT of GROUP BY CUBE or ROLLUP, where *nnn* is the number estimated by the IQ optimizer.

Set MAX_CUBE_RESULT to zero to override the default value. When this option is set to zero, the IQ optimizer does not check the row limit and allows the query to run. Setting MAX_CUBE_RESULT to zero is not recommended, as the query might not succeed.

MAX_CURSOR_COUNT Option

Specifies a resource governor to limit the maximum number of cursors that a connection can use at once.

Allowed Values

Integer

Default

50

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The specified resource governor allows a DBA to limit the number of cursors per connection that a user can have. If an operation exceeds the limit for a connection, an error is generated indicating that the limit has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the permissions of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

You can remove resource limits by setting MAX_CURSOR_COUNT to 0 (zero).

MAX_HASH_ROWS Option

Sets the maximum number of rows that the IQ optimizer considers for a hash algorithm.

Allowed Values

Integer from 1 to 4294967295

Default

2500000

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When generating a query plan, the IQ optimizer might have several algorithms (hash, sort, indexed) to choose from when processing a particular part of a query. These choices often depend on estimates of the number of rows to process or generate from that part of the query.

Database Options

This option sets an upper boundary for how many estimated rows are considered for a hash algorithm.

For example, if there is a join between two tables, and the estimated number of rows entering the join from both tables exceeds the value of `MAX_HASH_ROWS`, the optimizer does not consider a hash join. On systems with more than 50 MB per user of temporary buffer cache space, you might want to consider a higher value for this option.

MAX_IQ_THREADS_PER_CONNECTION Option

Controls the number of threads for each connection.

Allowed Values

3 – 10000

Default

144

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Allows you to constrain the number of threads (and thereby the amount of system resources) the commands executed on a connection use. For most applications, use the default.

MAX_IQ_THREADS_PER_TEAM Option

Controls the number of threads allocated to perform a single operation (such as a LIKE predicate on a column) executing within a connection.

Allowed Values

1 – 10000

Default

144

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Allows you to constrain the number of threads (and thereby the amount of system resources) allocated to a single operation. The total for all simultaneously executing teams for this connection is limited by the related option, MAX_IQ_THREADS_PER_CONNECTION. For most applications, use the default.

See also

- *MAX_IQ_THREADS_PER_CONNECTION Option* on page 578

MAX_JOIN_ENUMERATION Option

Controls the maximum number of tables to be optimized for join order after optimizer simplifications have been applied.

Allowed Values

1 – 64

Each **FROM** clause is limited to having at most 64 tables. In practice, however, the effective limit on the number of tables in a **FROM** clause is usually much lower, and is based partially on the complexity of the join relationships among those tables. That effective limit is constrained by the setting for **MAX_JOIN_ENUMERATION**. The optimizer will attempt to simplify the set of join relationships within a **FROM** clause. If those simplifications fail to reduce the set of the joins that must be simultaneously considered to no more than the current setting for **MAX_JOIN_ENUMERATION**, then the query will return an error.

Warning! Setting **MAX_JOIN_ENUMERATION** over the default value of 16 should only be done with caution, especially in the case of queries with bushy join relationships that can cause the amount of time required by the optimizer increase dramatically. In queries that use only a linear chain of join relationships, a **MAX_JOIN_ENUMERATION** setting of 64 can still provide reasonable optimization times.

Default

15

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Database Options

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The query optimizer simplifies its optimization of join order by separate handling of both lookup tables (that is, nonselective dimension tables) and tables that are effective Cartesian products. After simplification, it proceeds with optimizing the remaining tables for join order, up to the limit set by MAX_JOIN_ENUMERATION. If this limit is exceeded, the query is rejected with an error. The user can then either simplify the query or try increasing the limit.

Normally, you should not need to change this value. If you do, set MAX_JOIN_ENUMERATION as a temporary or user option.

MAX_PARTITIONED_HASH_MB Option

Sets an upper bound, expressed in megabytes, on the amount of temporary cache space that the optimizer can assume will be available for hash-partitioned hash-based query operators.

Allowed Values

Integer from 0 to 4294967295

Default

0

Scope

Can be set temporary for an individual connection, for a user, or for the PUBLIC group. No system privilege required to set this option. This option takes effect immediately.

Description

When generating a query plan, the IQ optimizer might choose from several algorithms when processing a particular part of a query. These decisions often depend on estimates of the temp cache space that will be required to process that part of the query and on the currently available temp cache. This option sets an upper bound on estimated temporary space usage for operators that can consider a hash-partitioned hash-based algorithm.

The default value of 0 indicates that there is no hard upper bound, and that therefore optimizer's choice will be limited only by the current temp cache availability, the current number of active user connections, and the HASH_PINNABLE_PERCENT option setting.

Note that this option affects only the optimizer's algorithm selection decisions, and that the run-time usage may under some circumstances occasionally exceed this limit.

MAX_PREFIX_PER_CONTAINS_PHRASE Option

Specifies the number of prefix terms allowed in a text search expression.

Allowed Values

0 – 300

Default

1

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Users must be licensed for the Unstructured Data Analytics Option to use **TEXT** indexes and perform full text searches.

MAX_QUERY_PARALLELISM Option

Sets upper bound for parallel execution of **GROUP BY** operations and for arms of a **UNION**.

Allowed Values

Integer less than, greater than or equal to number of CPUs.

Default

64

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This parameter sets an upper bound which limits how parallel the optimizer will permit query operators to go. This can influence the CPU usage for many query join, **GROUP BY**, **UNION**, **ORDER BY**, and other query operators.

Systems with more than 64 CPU cores often benefit from a larger value, up to the total number of CPU cores on the system to a maximum of 512; you can experiment to find the best value for this parameter for your system and queries.

Systems with 64 or fewer CPU cores should not need to reduce this value, unless excessive system time is seen. In that case, you can try reducing this value to determine if that adjustment can lower the CPU system time and improve query response times and overall system throughput.

MAX_QUERY_TIME Option

Sets a time limit so that the optimizer can disallow very long queries.

Allowed Values

0 to 2^{32} - 1 minutes

Default

0 (disabled)

Scope

Option can be set at the database (**PUBLIC**) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the **PUBLIC** value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the **SET ANY PUBLIC OPTION** system privilege to set this option. Can be set temporary for an individual connection or for the **PUBLIC** role. Takes effect immediately.

Description

If the query runs longer than the **MAX_QUERY_TIME** setting, SAP Sybase IQ stops the query and sends a message to the user and the IQ message file. For example:

```
The operation has been cancelled -- Max_Query_Time exceeded.
```

MAX_QUERY_TIME applies only to queries and not to any SQL statement that is modifying the contents of the database.

MAX_STATEMENT_COUNT Option

Specifies a resource governor to limit the maximum number of prepared statements that a connection can use at once.

Allowed Values

Integer

Default

100

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The specified resource governor allows a DBA to limit the number of prepared statements per connection that a user can have. If an operation exceeds the limit for a connection, an error is generated indicating that the limit has been exceeded.

If a connection executes a stored procedure, that procedure is executed under the permissions of the procedure owner. However, the resources used by the procedure are assigned to the current connection.

You can remove resource limits by setting MAX_STATEMENT_COUNT to 0 (zero).

MAX_TEMP_SPACE_PER_CONNECTION Option

Limits temporary store space used per connection.

Allowed Values

Integer (number of MB)

Default

0 (no limit on temporary store usage)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

Database Options

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY SYSTEM OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

By controlling space per connection, this option enables DBAs to manage the space for both loads and queries. If the connection exceeds the run time quota specified by `MAX_TEMP_SPACE_PER_CONNECTION`, SAP Sybase IQ rolls back the current statement and returns this message to the IQ message file or client user:

```
The current operation has been cancelled:  
Max_Temp_Space_Per_Connection exceeded
```

Conditions that may fill the buffer cache include read or write errors, lack of main or temp space, or being out of memory. SAP Sybase IQ may return the first error encountered in these situations and the DBA must determine the appropriate solution.

In a distributed query processing transaction, SAP Sybase IQ uses the values set for the `QUERY_TEMP_SPACE_LIMIT` and `MAX_TEMP_SPACE_PER_CONNECTION` options for the shared temporary store by limiting the total shared and local temporary space used by all nodes participating in the distributed query. This means that any single query cannot exceed the total temporary space limit (from `IQ_SYSTEM_TEMP` and `IQ_SHARED_TEMP` dbspaces), no matter how many nodes participate.

For example, if the limit is 100 and four nodes use 25 units of temporary space each, the query is within limits. If the sum of the total space used by any of the nodes exceeds 100, however, the query rolls back.

Examples

Set a 500GB limit for all connections:

```
SET OPTION  
PUBLIC.MAX_TEMP_SPACE_PER_CONNECTION = 512000
```

Set a 10TB limit for all connections:

```
SET OPTION  
PUBLIC.MAX_TEMP_SPACE_PER_CONNECTION = 10485760
```

Set a 5000MB limit for user wilson:

```
SET OPTION  
wilson.MAX_TEMP_SPACE_PER_CONNECTION = 5000
```

See also

- *QUERY_TEMP_SPACE_LIMIT Option* on page 615

MINIMIZE_STORAGE Option

Minimizes use of disk space for newly created columns in SAP Sybase IQ 15 databases.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Dependencies

MINIMIZE_STORAGE applies to databases running with

FP_NBIT_IQ15_COMPATIBILITY='ON'. If

FP_NBIT_IQ15_COMPATIBILITY='OFF', the database engine ignores this option.

Description

If FP_NBIT_IQ15_COMPATIBILITY='ON', MINIMIZE_STORAGE optimizes storage for new columns by using as little as one byte of disk space per row wherever appropriate. By default, this option is 'OFF' for the PUBLIC role, and the specialized storage optimization does not occur for all newly created columns; when MINIMIZE_STORAGE='OFF' for the PUBLIC role but 'ON' as a temporary user option, one-byte storage is used for new columns created by that user ID.

In SAP Sybase IQ 15.x databases, setting MINIMIZE_STORAGE=ON is equivalent to placing an **IQ UNIQUE 255** clause on every new column, with the exception of certain data types that are by nature too wide for one-byte storage. When MINIMIZE_STORAGE='ON', there is no need to specify **IQ UNIQUE**, except for columns with more than 65536 unique values.

When the ratio of main memory to the number of columns is large, turning MINIMIZE_STORAGE='ON' is beneficial. Otherwise, storage of new columns generally benefits by turning this option OFF.

Important: Avoid running a database when FP_NBIT_IQ15_COMPATIBILITY='ON'. All SAP Sybase IQ 15 runtime behavior is available with the SAP Sybase IQ 16.0 interface.

See also

- *FP_LOOKUP_SIZE Option* on page 535
- *INDEX_ADVISOR Option* on page 554
- *FP_LOOKUP_SIZE_PPM Option* on page 535
- *FP_NBIT_IQ15_COMPATIBILITY Option* on page 539

MIN_PASSWORD_LENGTH Option

Sets the minimum length for new passwords in the database.

Allowed Values

Integer greater than or equal to zero

The value is in bytes. For single-byte character sets, this is the same as the number of characters.

Default

3 characters

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description

This option imposes a minimum length on all new passwords for greater security. Existing passwords are not affected.

Example

Set the minimum length for new passwords to 6 bytes:

```
SET OPTION PUBLIC.MIN_PASSWORD_LENGTH = 6
```

MIN_ROLE_ADMINS Option

Configures the minimum number of required administrators for all roles.

Allowed Values

1 – 10

Default

1

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description

This options sets the minimum number of required administrators for all roles. This value applies to the minimum number of role administrators for each role, not the minimum number or role administrators for the total number of roles. When dropping roles or users, this value ensures that you never create a scenario where there are no users and roles left with sufficient system privilege to manage the remaining users and roles.

MONITOR_OUTPUT_DIRECTORY Option

Controls placement of output files for the IQ buffer cache monitor.

Allowed Values

String.

Default

Same directory as the database.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

MONITOR_OUTPUT_DIRECTORY controls the directory in which the IQ monitor output files are created, regardless of what is being monitored or what monitor mode is used. The dummy table used to start the monitor can be either a temporary or a permanent table. The directory can be on any physical machine.

All monitor output files are used for the duration of the monitor runs, which cannot exceed the lifetime of the connection. The output file still exists after the monitor run stops. A connection can run up to two performance monitors simultaneously, one for main cache and one for temp cache. A connection can run a monitor any number of times, successively.

The DBA can use the PUBLIC setting to place all monitor output in the same directory, or set different directories for individual users.

Example

This example shows how you could declare a temporary table for monitor output, set its location, and then have the monitor start sending files to that location for the main and temp buffer caches.

Note: In this example, the output directory string is set to both “/tmp” and “tmp/”. The trailing slash (“/”) is correct and is supported by the interface. The example illustrates that the buffer cache monitor does not require a permanent table; a temporary table can be used.

```
declare local temporary table dummy_monitor (dummy_column integer)
```

```
set option Monitor_Output_Directory = "/tmp"  
iq utilities main into dummy_monitor start monitor '-debug -interval  
2'
```

```
set option Monitor_Output_Directory = "tmp/"
```

```
iq utilities private into dummy_monitor start monitor '-debug -  
interval 2'
```

MPX_AUTOEXCLUDE_TIMEOUT Option

Timeout for autoexcluding a secondary node on the coordinator node. This option does not apply to the designated failover node.

Allowed Values

0 to 10080 minutes (1 week). 0 indicates that the nodes are not autoexcluded. Values must be exactly divisible by the `MPX_HEARTBEAT_FREQUENCY` setting in minutes. For example, if the `MPX_HEARTBEAT_FREQUENCY` setting is 120 (2 minutes), `MPX_AUTOEXCLUDE_TIMEOUT` must be divisible by 2.

Default

60 minutes

Scope

Option can be set at the database (PUBLIC) level only.

Requires the `SET ANY SYSTEM OPTION` system privilege to set this option. Setting takes effect immediately and persists across server restarts.

MPX_HEARTBEAT_FREQUENCY Option

Interval until the heartbeat thread wakes and performs periodic operations, such as checking for coordinator connectivity and cleaning up the connection pool on the secondary node. The

heartbeat thread maintains a dedicated internal connection from secondary server to coordinator.

Allowed Values

2 seconds to 3600 seconds

Default

60 seconds

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must restart the server for the change to take effect.

MPX_IDLE_CONNECTION_TIMEOUT Option

Time after which an unused connection in the connection pool on a secondary node will be closed.

Allowed Values

0 sec to no limit

Default

600 seconds

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Setting takes effect immediately and persists across server restarts.

MPX_LIVENESS_TIMEOUT Option

Time, in seconds, before a heartbeat on a secondary server declares the coordinator offline if the heartbeat fails to reconnect to the coordinator after the first disconnect. This option also determines how long the coordinator keeps a global transaction in a suspended state.

Allowed Values

0 to 604800 (1 week) in seconds

Default

3600 seconds (1 hour)

Scope

This option affects all multiplex nodes and has no node-specific or connection-specific value. Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. If you change the value of MPX_LIVENESS_TIMEOUT on a running server, the new value takes effect immediately for connections that might suspend in the future. The changed value also immediately affects the remaining timeout period for all current suspended transactions.

Description

If a writer fails to resume a suspended transaction within the MPX_LIVENESS_TIMEOUT period, the transaction can no longer commit, and the user should roll back the transaction. The coordinator keeps a global transaction in a suspended state for a period of $2 * MPX_LIVENESS_TIMEOUT$. If the corresponding writer fails to resume the transaction before the $2 * MPX_LIVENESS_TIMEOUT$ period, the coordinator rolls back the suspended transaction.

Always specify an MPX_LIVENESS_TIMEOUT value that is a multiple of the current MPX_HEARTBEAT_FREQUENCY value, which controls the aliveness check period. The coordinator internally doubles the value of MPX_LIVENESS_TIMEOUT.

MPX_MAX_CONNECTION_POOL_SIZE Option

Maximum number of connections allowed in the connection pool on a secondary node.

Allowed Values

1 to 1000

Default

10

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Setting takes effect immediately and persists across server restarts.

Description

INC connections are inter-server connections between secondary nodes and the coordinator node. An INC connection is associated with each user connection on a secondary server doing a DDL or read-write operation. The connection is active until that command commits or rolls back; it then returns to the pool. If these transactions are short lived, then the default setting of MPX_MAX_CONNECTION_POOL_SIZE suffices for many user connections running DDL or RW operations. If many concurrent connections run DDL or read-write operations, or the transactions take a long time, increase the value of MPX_MAX_CONNECTION_POOL_SIZE.

For example, increase the value when many user connections do concurrent loads without committing.

Exceeding `MPX_MAX_CONNECTION_POOL_SIZE` returns SQL Anywhere Error -1004000: The number of connections in the connection pool have exceeded the upper limit.

To estimate the pool size required, consider the setting of the `-gm` server option. The `-gm` setting indicates how many users can connect to the secondary server; the `INC` connections are not included, but will add to this number. Use application requirements to assess how many read-write or DDL operations are likely to occur per user, and increase the pool size accordingly.

Each connection (`INC` or `user`) carries a memory overhead depending on `-gn` setting and number of cores. The burden of memory and thread contention may affect SAP Sybase IQ server response times.

MPX_MAX_UNUSED_POOL_SIZE Option

Maximum number of unused connections in the connection pool on a secondary node.

Allowed Values

0 to maximum pool size

Default

0

Scope

Option can be set at the database (`PUBLIC`) level only.

Requires the `SET ANY SYSTEM OPTION` system privilege to set this option. Setting takes effect immediately and persists across server restarts.

MPX_WORK_UNIT_TIMEOUT Option

Time, in seconds, before a multiplex DQP leader reassigns incomplete distributed work to another DQP worker node.

Allowed Values

0 to 3600 seconds.

DQP work units are typically sized to span only a few seconds. If a worker node goes offline or experiences an unusually high workload, DQP work previously assigned to that worker node is reassigned to another node after the given timeout.

Default

60 seconds

Database Options

Typically you do not need to change this option from its default value. However, increase this option in rare cases where a query has very large intermediate results that cause individual work units to time out.

Decrease this option if unreliable networks or servers cause distributed work to be lost and the timeout interval is unacceptably long. Note that setting this option too low can cause unnecessary early timeouts.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

NEAREST_CENTURY Option [TSQL]

Controls the interpretation of 2-digit years, in string to date conversions.

Allowed Values

0 – 100

Default

50

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

NEAREST_CENTURY controls the handling of 2-digit years, when converting from strings to dates or timestamps.

The NEAREST_CENTURY setting is a numeric value that acts as a rollover point. Two-digit years less than the value are converted to 20yy, whereas years greater than or equal to the value are converted to 19yy.

Adaptive Server Enterprise and SAP Sybase IQ behavior is to use the nearest century, so that if the year value *yy* is less than 50, then the year is set to 20yy.

NOEXEC Option

Generates the optimizer query plans instead of executing the plan.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When determining how to process a query, the IQ optimizer generates a query plan to map how it plans to have the query engine process the query. If this option is set ON, the optimizer sends the plan for the query to the IQ message file rather than submitting it to the query engine. NOEXEC affects queries and commands that include a query.

Setting NOEXEC ON also prevents the execution of **INSERT...VALUES**, **INSERT...SELECT**, **INSERT...LOCATION**, **SELECT...INTO**, **LOAD TABLE**, **UPDATE**, **TRUNCATE TABLE**, **DELETE**, and updatable cursor operations.

When the `EARLY_PREDICATE_EXECUTION` option is ON, SAP Sybase IQ executes the local predicates for all queries before generating a query plan, even when the NOEXEC option is ON. The generated query plan is the same as the runtime plan.

See also

- *EARLY_PREDICATE_EXECUTION Option* on page 528

NON_ANSI_NULL_VARCHAR Option

Controls whether zero-length VARCHAR data is treated as NULLs for insert, load, and update operations.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

NON_ANSI_NULL_VARCHAR lets you revert to non-ANSI (Version 12.03.1) behavior for treating zero-length VARCHAR data during load or update operations. When this option is set to OFF, zero-length VARCHAR data is stored as zero-length during load, insert, or update. When this option is set to ON, zero-length VARCHAR data is stored as NULLs on load, insert, or update.

NON_KEYWORDS Option [TSQL]

Turns off individual keywords, allowing their use as identifiers.

Allowed Values

String

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

NON_KEYWORDS turns off individual keywords. If you have an identifier in your database that is now a keyword, you can either add double quotes around the identifier in all applications or scripts, or you can turn off the keyword using the NON_KEYWORDS option.

This statement prevents **TRUNCATE** and **SYNCHRONIZE** from being recognized as keywords:

```
SET OPTION NON_KEYWORDS = 'TRUNCATE, SYNCHRONIZE'
```

Each new setting of this option replaces the previous setting. This statement clears all previous settings:

```
SET OPTION NON_KEYWORDS =
```

A side effect of the options is that SQL statements using a turned-off keyword cannot be used; they produce a syntax error.

NOTIFY_MODULUS Option

Controls the default frequency of notify messages issued by certain commands.

Allowed Values

Any integer

Default

0 - For new SAP Sybase IQ 16.0 or later databases.

100000 - For pre-16.0 upgraded databases.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option sets the default number of notify messages SAP Sybase IQ issued for certain commands that produce them. The **NOTIFY** clause for some of the commands (such as **CREATE INDEX**, **LOAD TABLE**, and **DELETE**) override this value. Other commands that do not support the **NOTIFY** clause always use this value. The default does not restrict the number of messages you can receive.

ODBC_DISTINGUISH_CHAR_AND_VARCHAR Option

Controls how the Sybase IQ ODBC driver describes CHAR columns.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When a connection is opened, the Sybase IQ ODBC driver uses the setting of this option to determine how CHAR columns are described. If

ODBC_DISTINGUISH_CHAR_AND_VARCHAR is set to OFF (the default), then CHAR columns are described as SQL_VARCHAR. If this option is set to ON, then CHAR columns are described as SQL_CHAR. VARCHAR columns are always described as SQL_VARCHAR.

ON_CHARSET_CONVERSION_FAILURE Option

Controls the action taken, if an error is encountered during character conversion.

Allowed Values

Character Conversion Error	Action
IGNORE	Errors and warnings do not appear.
WARNING	Substitutions and illegal characters are reported as warnings. Illegal characters are not translated.
ERROR	Substitutions and illegal characters are reported as errors.

Default

IGNORE

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

ON_CHARSET_CONVERSION_FAILURE controls the action taken, if an error is encountered during character conversion.

Single-byte to single-byte converters are not able to report substitutions and illegal characters, and must be set to IGNORE.

ON_ERROR Option [Interactive SQL]

Controls the action taken if an error is encountered while executing statements in Interactive SQL.

Allowed Values

Value	Description
STOP	Interactive SQL stops executing statements from the file and returns to the statement window for input.
PROMPT	Interactive SQL prompts the user to see if he or she wants to continue.
CONTINUE	Errors display in the Messages pane and Interactive SQL continues executing statements.
EXIT	Interactive SQL terminates.
NOTIFY_CONTINUE	The error is reported, and the user is prompted to press Enter or click OK to continue.
NOTIFY_STOP	The error is reported, and the user is prompted to press Enter or click OK to stop executing statements.
NOTIFY_EXIT	The error is reported, and the user is prompted to press Enter or click OK to terminate Interactive SQL.

Default

PROMPT

Description

Controls the action taken, if an error is encountered while executing statements. When you are executing a .SQL file, the values STOP and EXIT are equivalent.

ON_TSQL_ERROR Option [TSQL]

Controls error handling in stored procedures.

Allowed Values

Value	Description
STOP	Stops execution immediately upon finding an error.
CONDITIONAL	If the procedure uses ON EXCEPTION RESUME , and the statement following the error handles the error, continue; otherwise, exit.
CONTINUE	Continue execution, regardless of the following statement. If there are multiple errors, the first error encountered in the stored procedure is returned. This option most closely mirrors Adaptive Server Enterprise behavior.

Default

CONDITIONAL

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

ON_TSQL_ERROR controls error handling in stored procedures.

Both CONDITIONAL and CONTINUE settings for ON_TSQL_ERROR are used for Adaptive Server Enterprise compatibility, with CONTINUE most closely simulating Adaptive Server Enterprise behavior. The CONDITIONAL setting is recommended, particularly when developing new Transact-SQL stored procedures, as CONDITIONAL allows errors to be reported earlier.

Adaptive Server Enterprise compatibility is described in *Reference: Building Blocks, Tables, and Procedures*.

When this option is set to **STOP** or **CONTINUE**, it supersedes the setting of the `CONTINUE_AFTER_RAISERROR` option. However, when this option is set to **CONDITIONAL** (the default), behavior following a **RAISERROR** statement is determined by the setting of the `CONTINUE_AFTER_RAISERROR` option.

See also

- *CREATE PROCEDURE Statement* on page 165
- *CREATE PROCEDURE Statement [T-SQL]* on page 172
- *RAISERROR Statement [T-SQL]* on page 381
- *CONTINUE_AFTER_RAISERROR Option [TSQL]* on page 498

OS_FILE_CACHE_BUFFERING Option

Controls use of file system buffering for IQ Main dbspaces.

Allowed Values

ON, OFF

Default

OFF; default affects newly created databases only.

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

Setting `OS_FILE_CACHE_BUFFERING OFF` prevents file system buffering for IQ Main Store files. Turning off file system buffering saves a data copy from the file system buffer cache to the main IQ buffer cache. Usually this reduces paging caused by competition for memory between the IQ buffer manager and the file system buffer of the operating system. When `OS_FILE_CACHE_BUFFERING` reduces paging, this option improves performance; however, if the IQ page size for the database is less than the block size of the file system (typically only in testing situations), performance decreases, especially during multiuser operation.

Experiment with this option to determine the best setting for different conditions. You must restart the database for the new setting to take effect.

This direct I/O performance option is available on Sun Solaris UFS, Linux, Linux IBM, AIX, and Windows file systems only. This option has no effect on HP-UX and HP-UXi and does not affect databases on raw disk. In Linux, direct I/O is supported in kernel versions 2.6.x.

To enable direct I/O on Linux kernel version 2.6 and AIX, also set the environment variable `IQ_USE_DIRECTIO` to 1. Direct I/O is disabled by default in Linux kernel version 2.6 and AIX. `IQ_USE_DIRECTIO` has no effect on Sun Solaris and Windows.

Note: SAP Sybase IQ does not support direct I/O on Linux kernel version 2.4. If you set the `IQ_USE_DIRECTIO` environment variable on Linux kernel version 2.4, the SAP Sybase IQ server does not start. The error `Error: Invalid Block I/O argument, maybe <pathname> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct IO on unsupported OS is reported.`

`OS_FILE_CACHE_BUFFERING_TEMPDB` controls file system buffering for IQ Temporary Store files.

See also

- *OS_FILE_CACHE_BUFFERING_TEMPDB Option* on page 600

OS_FILE_CACHE_BUFFERING_TEMPDB Option

Controls the use of file system buffering for IQ Temporary dbspaces.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must shut down and restart the database for the change to take effect.

Description

Setting `OS_FILE_CACHE_BUFFERING_TEMPDB` to OFF prevents file system buffering for IQ Temporary Store files. Turning off file system buffering saves a data copy from the file system buffer cache to the main IQ buffer cache. Usually this reduces paging caused by competition for memory between the IQ buffer manager and the file system buffer of the operating system. When `OS_FILE_CACHE_BUFFERING_TEMPDB` reduces paging, this option improves performance; however, if the IQ page size for the database is less than the block size of the file system (typically only in testing situations), performance decreases, especially during multiuser operation.

Experiment with this option to determine the best setting for different conditions. You must restart the database for the new setting to take effect.

This direct I/O performance option is available on Sun Solaris UFS, Linux, Linux IBM, AIX, and Windows file systems only. This option has no effect on HP-UX and HP-UXi and does not affect databases on raw disk. In Linux, direct I/O is supported in kernel versions 2.6.x.

To enable direct I/O on Linux kernel version 2.6 and AIX, also set the environment variable `IQ_USE_DIRECTIO` to 1. Direct I/O is disabled by default in Linux kernel version 2.6 and AIX. `IQ_USE_DIRECTIO` has no effect on Sun Solaris and Windows.

Note: SAP Sybase IQ

SAP Sybase IQ does not support direct I/O on Linux kernel version 2.4. If you set the `IQ_USE_DIRECTIO` environment variable on Linux kernel version 2.4, the SAP Sybase IQ server does not start. The error `Error: Invalid Block I/O argument, maybe <pathname> is a directory, or it exceeds maximum file size limit for the platform, or trying to use Direct IO on unsupported OS is reported.`

`OS_FILE_CACHE_BUFFERING` controls file system buffering for IQ Main Store files.

See also

- *OS_FILE_CACHE_BUFFERING Option* on page 599

POST_LOGIN_PROCEDURE Option

Specifies a login procedure whose result set contains messages that are displayed by the client application immediately after a user successfully logs in.

Allowed Values

String

Default

dbo.sa_post_login_procedure

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The default post login procedure, **dbo.sa_post_login_procedure**, executes immediately after a user successfully logs in.

If you have the SET ANY SECURITY OPTION system privilege, you can customize the post login actions by creating a new procedure and setting `POST_LOGIN_PROCEDURE` to call the new procedure. Do not edit **dbo.sa_post_login_procedure**. The customized post login procedure must be created in every database you use.

The post login procedure supports the client applications Interactive SQL, and Interactive SQL Classic.

See also

- *LOGIN_PROCEDURE Option* on page 572

PRECISION Option

Specifies the maximum number of digits in the result of any decimal arithmetic, for queries on the catalog store only.

Allowed Values

126

Default

126

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Takes effect immediately.

Description

Precision is the total number of digits to the left and right of the decimal point. The default PRECISION value is fixed at 126. The SCALE option specifies the minimum number of digits after the decimal point, when an arithmetic result is truncated to the maximum specified by PRECISION, for queries on the catalog store.

Note: In SQL Anywhere, the maximum value supported for the numeric function is 255. If the precision of the numeric function exceeds the maximum value supported, you see the error The result datatype for function '_funcname' exceeds the maximum supported numeric precision of 255. Please set the proper value for precision in numeric function, 'location'

See also

- *SCALE Option* on page 626
- *MAX_CLIENT_NUMERIC_PRECISION Option* on page 574

PREFETCH Option

Allows you to turn fetching on or off or to use the ALWAYS value to prefetch the cursor results, even for SENSITIVE cursor types and for cursors that involve a proxy table.

Allowed Values

ON, OFF, ALWAYS

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

For the catalog store only, PREFETCH controls whether rows are fetched to the client side before being made available to the client application. Fetching a number of rows at a time, even when the client application requests rows one at a time (for example, when looping over the rows of a cursor) minimizes response time and improves overall throughput by limiting the number of requests to the database.

The setting of PREFETCH is ignored by Open Client and JDBC connections, and for the IQ store.

PREFETCH_BUFFER_LIMIT Option

Specifies the amount of memory used for prefetching.

Allowed Values

Integer

Default

0

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must shut down the database and restart it for the change to take effect.

Description

`PREFETCH_BUFFER_LIMIT` defines the number of cache pages available to SAP Sybase IQ for use in prefetching (the read-ahead of database pages).

Do not set this option unless advised to do so by Sybase Technical Support.

See also

- *PREFETCH_BUFFER_PERCENT Option* on page 604

PREFETCH_BUFFER_PERCENT Option

Specifies the percent of memory used for prefetching.

Allowed Values

0 – 100

Default

40

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must shut down the database and restart it for the change to take effect.

Description

`PREFETCH_BUFFER_PERCENT` is an alternative to `PREFETCH_BUFFER_LIMIT`, as it specifies the percentage of cache available for use in prefetching.

Do not set this option unless advised to do so by Sybase Technical Support.

See also

- *PREFETCH_BUFFER_LIMIT Option* on page 603

PREFETCH_GARRAY_PERCENT Option

Specifies the percent of prefetch resources designated for inserts to **HG** indexes.

Allowed Values

0 – 100

Default

60

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

As with PREFETCH_SORT_PERCENT, this option designates a percentage of prefetch resources for use when inserting into an **HG** index.

Do not set this option unless advised to do so by Sybase Technical Support.

PREFETCH_SORT_PERCENT Option

Specifies the percent of prefetch resources designated for sorting objects.

Allowed Values

0 – 100

Default

20

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

PREFETCH_SORT_PERCENT designates a percentage of prefetch resources for use by a single sort object. Increasing this value can improve the single-user performance of inserts and deletes, but may have detrimental effects on multiuser operations.

Do not set this option unless advised to do so by Sybase Technical Support.

PRESERVE_SOURCE_FORMAT Option [database]

Controls whether the original source definition of procedures, views, and event handlers is saved in system files. If saved, the formatted source is saved in the column source in SYSTABLE, SYSPROCEDURE, and SYSEVENT.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

When PRESERVE_SOURCE_FORMAT is ON, the server saves the formatted source from **CREATE** and **ALTER** statements on procedures, views, and events, and puts original source definition in the source column of the appropriate system table.

Unformatted source text is stored in the same system tables, in the columns `proc_defn`, and `view_defn`. The formatted source column allows you to view the definitions with the spacing, comments, and case that you want.

This option can be turned off to reduce space used to save object definitions in the database. The option can be set only for the PUBLIC role.

QUERY_DETAIL Option

Specifies whether or not to include additional query information in the Query Detail section of the query plan.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When `QUERY_DETAIL` and `QUERY_PLAN` (or `QUERY_PLAN_AS_HTML`) are both turned on, SAP Sybase IQ displays additional information about the query when producing its query plan. When `QUERY_PLAN` and `QUERY_PLAN_AS_HTML` are OFF, this option is ignored.

When `QUERY_PLAN` is ON (the default), especially if `QUERY_DETAIL` is also ON, you might want to enable message log wrapping or message log archiving to avoid filling up your message log file.

See also

- *QUERY_PLAN Option* on page 608
- *QUERY_PLAN_AS_HTML Option* on page 609

QUERY_NAME Option

Gives a name to an executed query in its query plan.

Allowed Values

Quote-delimited string of up to 80 characters.

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

You can assign the `QUERY_NAME` option any quote-delimited string value, up to 80 characters. For example:

```
set temporary option Query_Name = 'my third query'
```

When this option is set, query plans that are sent to the `.iqmsg` file or `.html` file include a line near the top of the plan that looks like:

```
Query_Name: 'my third query'
```

If you set the option to a different value before each query in a script, it is much easier to identify the correct query plan for a particular query. The query name is also added to the file name for HTML query plans. This option has no other effect on the query.

QUERY_PLAN Option

Specifies whether or not additional query plans are printed to the SAP Sybase IQ message file.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When this option is turned ON, SAP Sybase IQ produces textual query plans in the IQ message file. These query plans display the query tree topography, as well as details about optimization and execution. When this option is turned OFF, those messages are suppressed. The information is sent to the <dbname>.iqmsg file.

See also

- *QUERY_DETAIL Option* on page 606
- *QUERY_PLAN_AFTER_RUN Option* on page 608
- *QUERY_PLAN_AS_HTML Option* on page 609

QUERY_PLAN_AFTER_RUN Option

Prints the entire query plan after query execution is complete.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When `QUERY_PLAN_AFTER_RUN` is turned ON, the query plan is printed after the query has finished running. This allows the query plan to include additional information, such as the actual number of rows passed on from each node of the query.

For this option to work, the `QUERY_PLAN` option must be set to ON. You can use this option in conjunction with `QUERY_DETAIL` to generate additional information in the query plan report.

See also

- *QUERY_DETAIL Option* on page 606
- *QUERY_PLAN Option* on page 608
- *QUERY_PLAN_AS_HTML Option* on page 609

QUERY_PLAN_AS_HTML Option

Generates graphical query plans in HTML format for viewing in a Web browser.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

QUERY_PLAN_AS_HTML causes graphical query plans to be generated in HTML format.

When you set this option, also set the QUERY_NAME option for each query, so you know which query is associated with the query plan.

SAP Sybase IQ writes the plans in the same directory as the .iqmsg file. Query plan file names follow these conventions:

user-name_query-name_server-type_server-number_YYYYMMDD_HHMMSS_query-number_fragment-number.html

For example, if the user DBA sets the temporary option QUERY_NAME to 'Query_1123', a file created on November 8, 2012 at exactly 8:30 a.m. is called DBA_QUERY_1123_L_0_20121108_083000_4.html. The date, time, and unique **query-number** appended to the file name ensure that existing files are not overwritten. The **server-type** parameter indicates whether the plan originates from a leader (**L**) or worker (**W**) node. The **server-number** identifies the server where the plan originated when all html files are routed to a single directory.

On multiplex servers, worker nodes generate an html file for each fragment executed by the worker, which can result in multiple html files from a single query. These files are identified by **fragment-number**.

Note: If you use this feature, monitor your disk space usage so you leave enough room for your .iqmsg and log files to grow. Enable IQ message log wrapping or message log archiving to avoid filling up your message log file.

QUERY_PLAN_AS_HTML acts independently of the setting for the QUERY_PLAN option. In other words, if QUERY_PLAN_AS_HTML is ON, you get an HTML format query plan whether or not QUERY_PLAN is ON.

This feature is supported with newer versions of many commonly used browsers. Some browsers might experience problems with plans generated for very complicated queries.

Simplex Output Example

Output generated from a query plan named Q1123 on a simplex server:

```
DBA_QUERY_Q1123_L_0_20121108_083000_4.html
```

Simplex servers always return a **server-type** parameter that indicates that the plan originated on a leader (**L**) with a **server-number** equal to 0. Simplex output never includes a **fragment-number**.

Multiplex Output Example

Output generated from a query plan named Q101 executed on a multiplex server. The leader node (**L**) **server-number** is 1; worker (**W**) nodes are numbered 2 and 3. Notice the **fragment-number** that appears after the query number in the worker output.

Output generated from the leader node:

```
DBA_L_1_Q101_20121108_083000_94.html
```

Output from one of the worker nodes:

```
DBA_W_2_Q101_20121108_083000_94_2.html
DBA_W_2_Q101_20121108_083000_94_1.html
```

Corresponding output from another worker:

```
DBA_W_3_Q101_20121113-054928_94_2.html
DBA_W_3_Q101_20121113-054933_94_1.html
```

See also

- *QUERY_NAME* Option on page 607
- *QUERY_PLAN* Option on page 608
- *QUERY_PLAN_AFTER_RUN* Option on page 608

QUERY_PLAN_AS_HTML_DIRECTORY Option

Specifies the directory into which SAP Sybase IQ writes the HTML query plans.

Allowed Values

String containing a directory path name

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When the `QUERY_PLAN_AS_HTML` option is turned ON and a directory is specified with the `QUERY_PLAN_AS_HTML_DIRECTORY` option, SAP Sybase IQ writes the HTML query plans in the specified directory. This option provides additional security by allowing HTML query plans to be produced outside of the server directory. When the

Database Options

`QUERY_PLAN_AS_HTML_DIRECTORY` option is not used, the query plans are sent to the default directory (the `.iqmsg` file directory).

If the `QUERY_PLAN_AS_HTML` option is ON and `QUERY_PLAN_AS_HTML_DIRECTORY` is set to a directory that does not exist, SAP Sybase IQ does not save the HTML query plan and no error is generated. In this case, the query continues to run and a message is logged to the IQ message file, so the DBA knows that the HTML query plan was not written. If the specified directory path or permissions on the directory are not correct, the message `Error opening HTML Query plan: filename` is written in the `.iqmsg` file.

Example

Create the example directory `/system1/users/DBA/html_plans` and set the correct permissions on the directory. Then set the options and run the query:

```
SET TEMPORARY OPTION QUERY_PLAN_AS_HTML = 'ON';
SET TEMPORARY OPTION QUERY_PLAN_AS_HTML_DIRECTORY = '/system1/users/DBA/html_plans';
SELECT col1 FROM tabl;
```

The HTML query plan is written to a file in the specified directory `/system1/users/DBA/html_plans`.

See also

- *QUERY_PLAN_AS_HTML Option* on page 609

QUERY_PLAN_MIN_TIME Option

Specifies a threshold for query execution. The post-query plan is generated only if query execution time exceeds the threshold.

Allowed Values

Integer, in milliseconds.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

A query with a very short execution time (a *micro query*) executes faster if a query plan is not generated. This option can be set to avoid the generation of query plans, and the associated query plan generation costs for these queries. The `QUERY_PLAN_MIN_TIME` option is ignored unless the following options are also set:

- `QUERY_PLAN = ON` or `QUERY_PLAN_AS_HTML = ON`
- `QUERY_PLAN_AFTER_RUN = ON`
- `QUERY_TIMING = ON`

When these options are set, setting a `QUERY_PLAN_MIN_TIME` query execution threshold prevents the generation of query plans for queries whose execution times do not exceed the specified threshold.

QUERY_PLAN_TEXT_ACCESS Option

Enables or prevents users from accessing query plans from the Interactive SQL client or from using SQL functions to get plans.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY SYSTEM OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

When `QUERY_PLAN_TEXT_ACCESS` option is `ON`, users can view, save, and print query plans from the Interactive SQL client. When the option is `OFF`, query plans are not cached, and other query plan-related database options have no affect on the query plan display from the Interactive SQL client. This error message displays:

```
No plan available. The database option QUERY_PLAN_TEXT_ACCESS is OFF.
```

See also

- *QUERY_DETAIL Option* on page 606
- *QUERY_PLAN_AFTER_RUN Option* on page 608

- *QUERY_PLAN_AS_HTML Option* on page 609
- *QUERY_PLAN_TEXT_CACHING Option* on page 614
- *OUTPUT Statement [Interactive SQL]* on page 371

QUERY_PLAN_TEXT_CACHING Option

Allows you to specify whether or not SAP Sybase IQ generates and caches IQ plans for queries executed by the user.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

IQ query plans vary in size and can become very large for complex queries. Caching plans for display on the Interactive SQL client can have high resource requirements. The `QUERY_PLAN_TEXT_CACHING` option gives users a mechanism to control resources for caching plans. With this option turned OFF (the default), the query plan is not cached for that user connection.

Note: If `QUERY_PLAN_TEXT_ACCESS` is turned OFF, the query plan is not cached for the connections from that user, no matter how `QUERY_PLAN_TEXT_CACHING` is set.

See also

- *QUERY_DETAIL Option* on page 606
- *QUERY_PLAN_AFTER_RUN Option* on page 608
- *QUERY_PLAN_AS_HTML Option* on page 609
- *QUERY_PLAN_TEXT_ACCESS Option* on page 613
- *OUTPUT Statement [Interactive SQL]* on page 371

QUERY_ROWS_RETURNED_LIMIT Option

Sets the row threshold for rejecting queries based on estimated size of result set.

Allowed Values

Any integer

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If SAP Sybase IQ receives a query that has an estimated number of result rows greater than the value of `QUERY_ROWS_RETURNED_LIMIT`, it rejects the query with this message:

```
Query rejected because it exceeds resource:
Query_Rows_Returned_Limit
```

If you set this option to zero (the default), there is no limit and no queries are ever rejected based on the number of rows in their output.

QUERY_TEMP_SPACE_LIMIT Option

Specifies the maximum estimated amount of temp space before a query is rejected.

Allowed Values

Any integer

Default

0 (no limit)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Database Options

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If SAP Sybase IQ receives a query that is estimated to require a temporary result space larger than value of this option, it rejects the query with this message:

```
Query rejected because it exceeds total space resource limit
```

When set to zero (the default), there is no limit on temporary store usage by queries.

Users may override this option in their own environments to run queries that can potentially fill up the entire temporary store. To prevent runaway queries from filling up the temporary store, a user with the SET ANY SYSTEM OPTION system privilege can set the option MAX_TEMP_SPACE_PER_CONNECTION. The MAX_TEMP_SPACE_PER_CONNECTION option monitors and limits actual temporary store usage for all DML statements, not just queries.

In a distributed query processing transaction, SAP Sybase IQ uses the values set for the QUERY_TEMP_SPACE_LIMIT and MAX_TEMP_SPACE_PER_CONNECTION options for the shared temporary store by limiting the total shared and local temporary space used by all nodes participating in the distributed query. This means that any single query cannot exceed the total temp space limit (from IQ_SYSTEM_TEMP and IQ_SHARED_TEMP dbspaces), no matter how many nodes participate.

For example, if the limit is 100 and four nodes use 25 units of temporary space each, the query is within limits. If the sum of the total space used by any of the nodes exceeds 100, however, the query rolls back.

See also

- *MAX_TEMP_SPACE_PER_CONNECTION Option* on page 583

QUERY_TIMING Option

Determines whether or not to collect specific timing statistics and display them in the query plan.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option controls the collection of timing statistics on subqueries and some other repetitive functions in the query engine.

Query timing is represented in the query plan detail as a series of timestamps. These timestamps correspond to query operator phases (Conditions, Prepare, Fetch, Complete). HTML and Interactive SQL query plans display query timing graphically as a timeline.

QUOTED_IDENTIFIER Option [TSQL]

Controls the interpretation of strings that are enclosed in double quotes.

Allowed Values

ON, OFF

Default

ON

OFF for Open Client connections.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

QUOTED_IDENTIFIER controls whether strings enclosed in double quotes are interpreted as identifiers (ON) or as literal strings (OFF). This option is included for Transact-SQL compatibility.

Sybase Control Center and Interactive SQL set QUOTED_IDENTIFIER temporarily to ON, if it is set to OFF. A message is displayed informing you of this change. The change is in effect only for the Sybase Control Center or Interactive SQL connection. The JDBC driver also temporarily sets QUOTED_IDENTIFIER to ON.

RECOVERY_TIME Option

Sets the maximum length of time, in minutes, that the database server takes to recover from system failure.

Allowed Values

Integer, in minutes

Default

2

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must restart the server for the change to take effect.

Description

Use this option with the CHECKPOINT_TIME option to decide when checkpoints should be done.

A heuristic measures the recovery time based on the operations since the last checkpoint. Thus, the recovery time is not exact.

See also

- *CHECKPOINT_TIME Option* on page 496

RESERVED_KEYWORDS Option

Turns on individual keywords that are disabled by default.

Allowed Values

String

Default

Empty string

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

This option turns on individual keywords that are disabled by default. Only the LIMIT keyword can be turned on.

Examples

The following statement allows the LIMIT keyword to be recognized as a keyword:

```
SET OPTION RESERVED_KEYWORDS = 'LIMIT';
```

You cannot turn on the keywords SET, OPTION, and OPTIONS. The following determine whether a word is identified as a keyword (in order of precedence):

- It appears in the SQL Anywhere list of reserved words
- It is turned on with the RESERVED_KEYWORDS option
- It is turned off with the NON_KEYWORDS option

Each setting of this option replaces the previous setting. The following statement clears all previous settings:

```
SET OPTION RESERVED_KEYWORDS = ;
```

RETURN_DATE_TIME_AS_STRING Option

Controls how a date, time, or timestamp value is passed to the client application when queried.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set as a temporary option only, for the duration of the current connection or for the PUBLIC role.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Takes effect immediately.

Description

RETURN_DATE_TIME_AS_STRING indicates whether date, time, and timestamp values are returned to applications as a date or time data type or as a string.

When this option is set to ON, the server converts the date, time, or timestamp value to a string before it is sent to the client in order to preserve the `TIMESTAMP_FORMAT`, `DATE_FORMAT`, or `TIME_FORMAT` option setting.

Sybase Control Center and Interactive SQL automatically turn the RETURN_DATE_TIME_AS_STRING option ON.

See also

- *DATE_FORMAT Option* on page 512
- *TIME_FORMAT Option* on page 654

- *TIMESTAMP_FORMAT* Option on page 655

REVERT_TO_V15_OPTIMIZER Option

Setting this option ON forces the query optimizer to mimic SAP Sybase IQ 15.x behavior.

Allowed Values

ON, OFF

Default

- ON in all 16.0 databases upgraded from 15.x
- OFF in all newly created 16.0 databases

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. If permitted, can be set for an arbitrary other user or role, or for all users via the role. Takes effect immediately.

Description

SAP Sybase IQ 16.0 supports several new join and grouping algorithms that leverage Hash and Hash-Range partitioned tables, as well as a few other new algorithms. By default, all of these new algorithms are considered by the optimizer and will be selected where valid and appropriate. Setting `REVERT_TO_V15_OPTIMIZER='ON'` disables all 16.0 changes to the optimizer cost models. It also disables all of these new join and grouping algorithms, unless they are valid and are specifically requested via a positive value for either the `AGGREGATION_PREFERENCE` option, the `JOIN_PREFERENCE` option, or a join condition hint string.

The `REVERT_TO_V15_OPTIMIZER` option is normally used for internal testing and manually tuning queries. Only experienced DBAs should use it.

Note: An error will result if your query references an RLV-enabled table and `REVERT_TO_V15_OPTIMIZER='ON'`.

ROUND_TO_EVEN Option

Controls behavior of the SQL function **ROUND**.

Allowed Values

ON, OFF

Default

OFF

Scope

Requires the SET ANY SYSTEM OPTION system privilege. Can be set for the PUBLIC role only. Takes effect immediately.

Description

When ROUND_TO_EVEN option is set to ON, the **ROUND** function rounds half to the nearest even number. When the option is set to OFF, the **ROUND** function rounds SAP Sybase IQ half away from zero.

Sybase Control Center and Interactive SQL automatically turn the ROUND_TO_EVEN option OFF.

The **ROUND** function returns a different value based on whether ROUND_TO_EVEN is ON or OFF.

When ROUND_TO_EVEN is set ON, `SELECT ROUND (convert(double, 0.25), 1) from iq_dummy` returns 0.2.

When ROUND_TO_EVEN is set OFF, `SELECT ROUND (convert(double, 0.25), 1) from iq_dummy` returns 0.3.

ROW_COUNT Option

Limits the number of rows returned from a query.

Allowed Values

Integer.

Default

0 (no limit on rows returned)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When this runtime option is set to a nonzero value, query processing stops after the specified number of rows.

This option affects only statements with the keyword **SELECT** and does not affect **UPDATE** and **DELETE** statements.

The **SELECT** statement keywords **FIRST** and **TOP** also limit the number of rows returned from a query. Using **FIRST** is the same as setting the `ROW_COUNT` database option to 1. Using **TOP** is the same as setting `ROW_COUNT` to the same number of rows. If both **TOP** and `ROW_COUNT` are set, then the value of **TOP** takes precedence.

The `ROW_COUNT` option could produce non-deterministic results when used in a query involving global variables, system functions or proxy tables. Such queries are partly executed using CIS (Component Integrated Services). In such cases, use **SELECT TOP *n*** instead of setting `ROW_COUNT`, or set the global variable to a local one and use that local variable in the query.

See also

- *QUERY_ROWS_RETURNED_LIMIT* Option on page 615
- *SELECT Statement* on page 417

RV_AUTO_MERGE_EVAL_INTERVAL Option

This option configures the evaluation period used to determine when an automated merge of the row-level versioned (RLV) and IQ main stores should occur.

Allowed Values

1 – MAX_UINT (minutes)

Default

15 (minutes)

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

This option is used to configure the period of wait time, in minutes, between activations of the merge evaluator. The merge evaluator examines the merge parameters of each row-level versioning (RLV) enabled table against configured threshold values to determine whether a non-blocking (background) merge of the RLV table to IQ main stores should occur.

If the interval ends while the evaluator is active, or when a merge is already in progress, the interval re-sets.

Any new value for the interval is used when the merge evaluator is next activated.

RV_MAX_ACTIVE_SUBFRAGMENT_COUNT Option

This value maximizes utilization of the number of cores on the machine.

Allowed Values

≥ 0

Default

0

Note: Use of any value other than the default is not recommended as it could negatively impact CPU utilization and scalability of bulk loads.

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. This option takes effect on a table with the first write operation on the table. If the value of this option is changed after the first write operation has occurred, the new value does not take effect on the table until after a restart of the server.

Description

If the value is set to anything other than the default, the system uses the specified value or the total number of cores on the machine, whichever is less.

RV_MERGE_NODE_MEMSIZE Option

An automated merge of the row-level versioned (RLV) store and IQ main stores occurs based on the merge thresholds, including RV_MERGE_NODE_MEMSIZE. When this node threshold is exceeded, a merge will be triggered.

Allowed Values

0 - 100 (percent)

Default

75 (percent)

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

This option sets the percentage of total RLV memory size as a merge threshold for the node. If the total RLV memory size surpasses the threshold, the merge condition evaluator will determine which table(s) to merge. If multiple tables must be merged to satisfy the node threshold, parallel merges will be triggered for each table to be merged.

RV_MERGE_TABLE_MEMPERCENT Option

An automated merge of the row-level versioned (RLV) store and IQ main stores occurs based on the merge thresholds, including RV_MERGE_TABLE_MEMPERCENT. If this table threshold is exceeded, a merge will be triggered for the specific table.

Allowed Values

0 - 100 (percent)

Default

0 (percent)

Note: When RV_MERGE_TABLE_MEMPERCENT = 0, then the system uses a (per-table) threshold of $100\% / N$, where N is the number of RLV-enabled tables that have been loaded.

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

This option sets the percentage of memory used as a merge threshold for an RLV-enabled table. If the memory used surpasses the threshold, a merge will occur.

The system evaluates whether to merge the RLV and IQ main stores on a per-table basis. It enumerates through all loaded RLV tables, and for each one decides whether a merge is warranted. A merge for a single table is deemed warranted if:

1. The table violates either the memory threshold (RV_MERGE_TABLE_MEMPERCENT) or the row threshold (RV_MERGE_TABLE_NUMROWS), and
2. The system does not determine that a large percentage of the RLV rows are uncommitted, and are therefore unable to be merged.

RV_MERGE_TABLE_NUMROWS Option

An automated merge of the row-level versioned (RLV) store and IQ main stores occurs based on the merge thresholds, including `RV_MERGE_TABLE_NUMROWS`. If this table threshold is exceeded, a merge will be triggered for the specific table.

Allowed Values

1000 - 100000000

Default

10000000

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

This option sets the number of rows used as a merge threshold for an RLV-enabled table. If the number of rows used surpasses the threshold, a merge will occur.

The system evaluates whether to merge the RLV and IQ main stores on a per-table basis. It enumerates through all loaded RLV tables, and for each one decides whether a merge is warranted. A merge for a single table is deemed warranted if:

1. The table violates either the memory threshold (`RV_MERGE_TABLE_MEMPERCENT`) or the row threshold (`RV_MERGE_TABLE_NUMROWS`), and
2. The system does not determine that a large percentage of the RLV rows are uncommitted, and are therefore unable to be merged.

RV_RESERVED_DBSPACE_MB Option

A portion of the RLV store must be reserved for memory used by data structures during critical operations.

Allowed Values

Integer greater than or equal to 50 (megabytes)

Default

lesser of 50 Mb or half the size of the RLV dbspace

Scope

Option can be set at the database (PUBLIC) level only.

Database Options

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately. The server does not need to be restarted in order to change reserved space size.

Description

This option allows you to control the amount of space set aside in the RLV store for small but critical data structures used during release savepoint, commit, and rollback operations.

SCALE Option

Specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION, for queries on the catalog store only.

Allowed Values

Integer, with a maximum of 126.

Default

38

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Takes effect immediately.

Description

This option specifies the minimum number of digits after the decimal point when an arithmetic result is truncated to the maximum PRECISION, for queries on the catalog store.

Multiplication, division, addition, subtraction, and aggregate functions may all have results that exceed the maximum precision.

See also

- *MAX_CLIENT_NUMERIC_SCALE Option* on page 575
- *PRECISION Option* on page 602

SNAPSHOT_VERSIONING Option

Applies to RLV-enabled tables only (as opposed to all base tables in the database). Controls whether RLV-enabled tables are accessed using single-writer table-level versioning, or multiple writer row-level versioning. This option does not apply to the IQ catalog store.

Allowed Values

Value	Action
row-level	<p>Enables concurrent writer access and row-level versioning for RLV-enabled tables.</p> <p>The first transaction to modify a table row establishes a row write lock that persists until the end of the transaction.</p> <p>Subsequent transactions attempting to modify a locked row either fail with a lock/future version error, or block until the lock is released based on the value of the BLOCKING option.</p>
table-level	<p>Enables single-writer access and table-level versioning.</p> <p>The first transaction to access the table establishes a table write lock which persists until the end of the transaction.</p> <p>Subsequent transactions attempting to write to a locked table either fail with a lock/future version error, or block until the lock is released based on the value of the BLOCKING option.</p>

Default

table-level

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Takes effect immediately.

SIGNIFICANTDIGITSFORDOUBLEEQUALITY Option

Specifies the number of significant digits to the right of the decimal in exponential notation that are used in equality tests between two complex arithmetic expressions.

Allowed Values

0 – 15

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Because doubles are stored in binary (base 2) instead of decimal (base 10), this setting gives the approximate number of significant decimal digits used. If set to 0, all digits are used.

For example, when SIGNIFICANTDIGITSFORDOUBLEEQUALITY is set to 12, these numbers compare as equal; when set to 13, they do not:

- 1.23456789012345
- 1.23456789012389

SIGNIFICANTDIGITSFORDOUBLEEQUALITY affects equality tests between two complex arithmetic expressions, not those done by the indexes.

SORT_COLLATION Option

Allows implicit use of the **SORTKEY** function on **ORDER BY** expressions.

Allowed Values

Internal, collation_name, or collation_id

Default

Internal

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When the value of SORT_COLLATION is Internal, the **ORDER BY** clause remains unchanged.

When the value of this option is set to a valid collation name or collation ID, any string expression in the **ORDER BY** clause is treated as if the **SORTKEY** function has been invoked.

Functions are described in *Reference: Building Blocks, Tables, and Procedures*.

Example

Set the sort collation to binary:

```
SET TEMPORARY OPTION sort_collation='binary';
```

Setting the sort collation to binary transforms these queries:

```
SELECT Name, ID
FROM Products
ORDER BY Name, ID;
SELECT Name, ID
FROM Products
ORDER BY 1, 2;
```

The queries are transformed into:

```
SELECT Name, ID
FROM Products
ORDER BY SORTKEY(Name, 'binary'), ID;
```

SORT_PINNABLE_CACHE_PERCENT Option

Specifies the maximum percentage of currently available buffers a sort object tries to pin.

Allowed Values

0 – 100

Default

20

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

Database Options

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

For very large sorts, a larger value might help reduce the number of merge phases required by the sort. A larger number, however, might impact the sorts and hashes of other users running on the system. If you change this option, experiment to find the best value to increase performance, as choosing the wrong value might decrease performance.

Tip: Use the default value for SORT_PINNABLE_CACHE_PERCENT.

This option is primarily for use by Sybase Technical Support. If you change the value of SORT_PINNABLE_CACHE_PERCENT, do so with extreme caution.

SQL_FLAGGER_ERROR_LEVEL Option [TSQL]

Controls the behavior in response to any SQL code that is not part of the specified standard.

Allowed Values

- OFF
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Flags as an error any SQL code that is not part of a specified standard. For example, specifying SQL:2003/Package causes the database server to flag syntax that is not full SQL/2003 syntax.

For compatibility with previous SAP Sybase IQ versions, the values in this table are also accepted, and are mapped as specified.

Table 18. SQL_FLAGGER_ERROR_LEVEL Compatibility Values

Value	Action
E	Flag syntax that is not entry-level SQL92 syntax. Corresponds to SQL:1992/Entry.
I	Flag syntax that is not intermediate-level SQL92 syntax. Corresponds to SQL:1992/Intermediate.
F	Flag syntax that is not full-SQL92 syntax. Corresponds to SQL:1992/Full.
W	Allow all supported syntax. Corresponds to OFF.

SQL_FLAGGER_WARNING_LEVEL Option [TSQL]

Controls the response to any SQL that is not part of the specified standard.

Allowed Values

- OFF
- SQL:1992/Entry
- SQL:1992/Intermediate
- SQL:1992/Full
- SQL:1999/Core
- SQL:1999/Package
- SQL:2003/Core
- SQL:2003/Package

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Flags as an error any SQL code that is not part of a specified standard as a warning. For example, specifying SQL:2003/Package causes the database server to flag syntax that is not full SQL/2003 syntax.

The default behavior, OFF, turns warning flagging off.

For compatibility with previous SAP Sybase IQ versions, the values in this table are also accepted, and are mapped as specified.

Table 19. SQL_FLAGGER_WARNING_LEVEL Compatibility Values

Value	Action
E	Flag syntax that is not entry-level SQL92 syntax. Corresponds to SQL:1992/Entry.
I	Flag syntax that is not intermediate-level SQL92 syntax. Corresponds to SQL:1992/Intermediate.
F	Flag syntax that is not full-SQL92 syntax. Corresponds to SQL:1992/Full.
W	Allow all supported syntax. Corresponds to OFF.

STRING_RTRUNCATION Option [TSQL]

Determines whether an error is raised when an **INSERT** or **UPDATE** truncates a CHAR or VARCHAR string.

Allowed Values

ON, OFF

Default

ON

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If the truncated characters consist only of spaces, no exception is raised. ON corresponds to SQL92 behavior. When STRING_RTRUNCATION is OFF, the exception is not raised and the

character string is silently truncated. If the option is ON and an error is raised, a **ROLLBACK** occurs.

This option was OFF by default prior to SAP Sybase IQ 15.0. It can safely be set to OFF for backward compatibility. However, the ON setting is preferable to identify statements where truncation may cause data loss.

SUBQUERY_CACHING_PREFERENCE Option

Controls which algorithm to use for processing correlated subquery predicates.

Allowed Values

Value	Action
1	Use sort-based processing for the first subquery predicate. Other subquery predicates that do not have the same ordering key are processed using a hash table to cache subquery results.
2	Use the hash table to cache results for all subquery predicates when it is legal. If available temp cache cannot accommodate all of the subquery results, performance may be poor.
3	Cache one previous subquery result. Does not use SORT and HASH .
0	Let the optimizer choose.
-1	Avoid using SORT . The IQ optimizer chooses HASH if it is legal.
-2	Avoid using HASH . The IQ optimizer chooses SORT or cache-one value if it is legal.
-3	Avoid using cache-one value. The IQ optimizer chooses either HASH or SORT if it is legal.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

For correlated subquery predicates, the IQ optimizer offers a choice of caching outer references and subquery results that reduces subquery execution costs.

SUBQUERY_CACHING_PREFERENCE lets you override the optimizer's costing decision when choosing which algorithm to use. It does not override internal rules that determine whether an algorithm is legal within the query engine.

A setting of a non-zero value affects every subquery predicate in the query. A non-zero value cannot be used selectively for one subquery predicate in a query.

SUBQUERY_CACHING_PREFERENCE is normally used for internal testing by experienced DBAs only. It does not apply to **IN** subqueries.

See also

- *IN_SUBQUERY_PREFERENCE Option* on page 552

SUBQUERY_FLATTENING_PERCENT Option

Allows the user to change the threshold at which the optimizer decides to transform scalar subqueries into joins.

Allowed Values

Value	Action
0	The optimizer cost model decides
1 to (2 ³² -1)	The percentage of references at which to flatten

Default

100

Scope

This option only applies to correlated scalar subqueries. Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately. If you set SUBQUERY_FLATTENING_PERCENT to a non-default value, every scalar subquery predicate in the query is affected; this option cannot be used selectively for one scalar subquery predicate in a query.

Description

The SAP Sybase IQ query optimizer can convert a correlated scalar subquery into an equivalent join operation to improve query performance. The `SUBQUERY_FLATTENING_PERCENT` option allows the user to adjust the threshold at which this optimization occurs.

`SCALAR_FLATTENING_PERCENT` represents a percent of estimated inner distinct values to estimated outer distinct values in a scalar subquery. As the estimated percent approaches 100%, the cost of evaluating the subquery as a join is likely to be smaller than using individual index probes. The value may be set larger than 100%, since the estimated inners are not guaranteed to be less than estimated outers.

See also

- *SUBQUERY_FLATTENING_PREFERENCE* Option on page 635

SUBQUERY_FLATTENING_PREFERENCE Option

Allows a user to override the decisions of the optimizer when transforming (flattening) scalar or **EXISTS** subqueries into joins.

Allowed Values

Value	Action
-3	Avoid flattening both EXISTS and scalar subqueries to a join operation.
-2	Avoid flattening a scalar subquery to a join operation.
-1	Avoid flattening an EXISTS subquery to a join operation.
0	Allow the IQ optimizer to decide to flatten subqueries.
1	Ignore cost flattening EXIST , if possible.
2	Ignore cost flattening scalar, if possible.
3	Ignore cost of both EXISTS and scalar subquery.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Database Options

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately. If you set the option to a non-zero value, every subquery predicate in the query is affected; this option cannot be used selectively for one subquery predicate in a query.

Description

The SAP Sybase IQ optimizer may convert a correlated scalar subquery or an **EXISTS** or **NOT EXISTS** subquery into an equivalent join operation to improve query performance. This optimization is called *subquery flattening*. SUBQUERY_FLATTENING_PREFERENCE allows you to override the costing decision of the optimizer when choosing the algorithm to use.

Setting SUBQUERY_FLATTENING_PREFERENCE to 0 (allow the IQ optimizer to decide to flatten subqueries) is equivalent to setting the now deprecated FLATTEN_SUBQUERIES option to ON in earlier versions of SAP Sybase IQ.

See also

- *SUBQUERY_FLATTENING_PERCENT* Option on page 634

SUBQUERY_PLACEMENT_PREFERENCE Option

Controls the placement of correlated subquery predicate operators within a query plan.

Allowed Values

Value	Action
-1	Prefer the lowest possible location in the query plan, thereby placing the execution of the subquery as early as possible within the query.
0	Let the optimizer choose.
1	Prefer the highest possible location in the query plan, thereby delaying the execution of the subquery to as late as possible within the query.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

For correlated subquery operators within a query, the IQ optimizer may have a choice of several different valid locations within that query's plan.

`SUBQUERY_PLACEMENT_PREFERENCE` allows you to override the optimizer's cost-based decision when choosing the placement location. It does not override internal rules that determine whether a location is valid, and in some queries, there might be only one valid choice. If you set this option to a nonzero value, it affects every correlated subquery predicate in a query; it cannot be used to selectively modify the placement of one subquery out of several in a query.

This option is normally used for internal testing, and only experienced DBAs should use it.

The default setting of this option is almost always appropriate. Occasionally, Sybase Technical Support might ask you to change this value.

SUPPRESS_TDS_DEBUGGING Option

Determines whether TDS debugging information appears in the server window.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When the server is started with the `-z` option, debugging information appears in the server window, including debugging information about the TDS protocol.

`SUPPRESS_TDS_DEBUGGING` restricts the debugging information about TDS that appears in the server window. When this option is set to OFF (the default), TDS debugging information appears in the server window.

SWEEPER_THREADS_PERCENT Option

Specifies the percentage of SAP Sybase IQ threads used to sweep out buffer caches.

Allowed Values

1 – 40

Default

10

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must shut down the database and restart it for the change to take effect.

Description

SAP Sybase IQ uses a small percentage of its processing threads as sweeper threads. These sweeper threads clean out dirty pages in the main and temp buffer caches.

In the IQ Monitor **-cache** report, the GDirty column shows the number of times the LRU buffer was grabbed in a “dirty” (modified) state. If GDirty is greater than 0 for more than a brief time, you might need to increase SWEEPER_THREADS_PERCENT or WASH_AREA_BUFFERS_PERCENT.

The default setting of this option is almost always appropriate. Occasionally, SAP Sybase Technical Support might ask you to increase this value.

See also

- *WASH_AREA_BUFFERS_PERCENT Option* on page 662

TDS_EMPTY_STRING_IS_NULL Option [database]

Controls whether empty strings are returned as NULL or a string containing one blank character for TDS connections.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required

to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

TDS_EMPTY_STRING_IS_NULL is set to OFF by default and empty strings are returned as a string containing one blank character for TDS connections. When this option is set to ON, empty strings are returned as NULL strings for TDS connections. Non-TDS connections distinguish empty strings from NULL strings.

TEMP_EXTRACT_APPEND Option

Specifies that any rows extracted by the data extraction facility are added to the end of an output file.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option specifies that any rows extracted by the data extraction facility are added to the end of an output file. You create the output file in a directory where you have WRITE/EXECUTE permissions and you set WRITE permission on the directory and output file for the user name used to start SAP Sybase IQ (for example, sybase). You can give permissions on the output file to other users as appropriate. The name of the output file is specified in the TEMP_EXTRACT_NAME1 option. The data extraction facility creates the output file, if the file does not already exist.

TEMP_EXTRACT_APPEND is not compatible with the TEMP_EXTRACT_SIZE_n options. If you try to restrict the size of the extract append output file, SAP Sybase IQ reports an error.

See also

- *TEMP_EXTRACT_NAME_n Options* on page 643

TEMP_EXTRACT_BINARY Option

In combination with the `TEMP_EXTRACT_SWAP` option, specifies the type of extraction performed by the data extraction facility.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporarily for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Use this option with the `TEMP_EXTRACT_SWAP` option to specify the type of extraction performed by the data extraction facility.

Table 20. Extraction Option Settings for Extraction Type

Extraction type	TEMP_EXTRACT_BINARY	TEMP_EXTRACT_SWAP
binary	ON	OFF
binary/swap	ON	ON
ASCII	OFF	OFF

The default extraction type is ASCII.

See also

- *TEMP_EXTRACT_SWAP Option* on page 651

TEMP_EXTRACT_COLUMN_DELIMITER Option

Specifies the delimiter between columns in the output of the data extraction facility for an ASCII extraction.

Allowed Values

String

Default

','

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Use TEMP_EXTRACT_COLUMN_DELIMITER to specify the delimiter between columns in the output of the data extraction facility. In the case of an ASCII extraction, the default is to separate column values with commas. Strings are unquoted by default.

The delimiter must occupy 1 – 4 bytes, and must be valid in the collation order you are using, if you are using a multibyte collation order. Choose a delimiter that does not occur in any of the data output strings themselves.

If you set this option to the empty string '' for ASCII extractions, the extracted data is written in fixed-width ASCII with no column delimiter. Numeric and binary data types are right-justified on a field of *n* blanks, where *n* is the maximum number of bytes needed for any value of that type. Character data types are left-justified on a field of *n* blanks.

Note: The minimum column width in a fixed-width ASCII extraction is 4 bytes to allow the string "NULL" for a NULL value. For example, if the extracted column is CHAR(2) and TEMP_EXTRACT_COLUMN_DELIMITER is set to the empty string '', there are two spaces after the extracted data.

See also

- *TEMP_EXTRACT_QUOTE Option* on page 647
- *TEMP_EXTRACT_QUOTES Option* on page 648
- *TEMP_EXTRACT_QUOTES_ALL Option* on page 648
- *TEMP_EXTRACT_ROW_DELIMITER Option* on page 649

TEMP_EXTRACT_DIRECTORY Option

Controls whether a user is allowed to use the data extraction facility. Also controls the directory into which temp extract files are placed and overrides a directory path specified in the `TEMP_EXTRACT_NAMEn` options.

Allowed Values

string

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Can be set temporarily for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If the `TEMP_EXTRACT_DIRECTORY` option is set to the string FORBIDDEN (case insensitive) for a user, then that user is not allowed to perform data extracts. An attempt by this user to use the data extraction facility results in the error: You do not have permission to perform Extracts.

If `TEMP_EXTRACT_DIRECTORY` is set to FORBIDDEN for the PUBLIC role, then no one can run data extraction.

If `TEMP_EXTRACT_DIRECTORY` is set to a valid directory path, temp extract files are placed in that directory, overriding a path specified in the `TEMP_EXTRACT_NAMEn` options.

If `TEMP_EXTRACT_DIRECTORY` is set to an invalid directory path, an error occurs: Files does not exist File: <invalid path>

If `TEMP_EXTRACT_DIRECTORY` is blank, then temp extract files are placed in directories according to their specification in `TEMP_EXTRACT_NAMEn`. If no path is specified as part of `TEMP_EXTRACT_NAMEn`, the extract files are by default placed in the server startup directory.

This option provides increased security and helps control disk management by restricting the creation of large data extraction files to the directories for which a user has write access.

For details on the data extraction facility and using the extraction options, see *Administration: Load Management*.

See also

- *TEMP_EXTRACT_NAME*n Options on page 643

TEMP_EXTRACT_ESCAPE_QUOTES Option

Specifies whether all quotes in fields containing quotes are escaped in the output of the data extraction facility for an ASCII extraction.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option is ignored unless **TEMP_EXTRACT_QUOTE** is the default or set to the value of "" (double quotes), and **TEMP_EXTRACT_BINARY** is OFF, and either **TEMP_EXTRACT_QUOTES** or **TEMP_EXTRACT_QUOTES_ALL** is ON.

See also

- *TEMP_EXTRACT_BINARY* Option on page 640
- *TEMP_EXTRACT_QUOTES* Option on page 648
- *TEMP_EXTRACT_QUOTES_ALL* Option on page 648

TEMP_EXTRACT_NAMEn Options

Specifies the names of the output files or named pipes used by the data extraction facility. There are eight options: **TEMP_EXTRACT_NAME1** through **TEMP_EXTRACT_NAME8**.

Allowed Values

string

Default

" (the empty string)

Scope

Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role.

Description

TEMP_EXTRACT_NAME1 through TEMP_EXTRACT_NAME8 specify the names of the output files used by the data extraction facility. You must use these options sequentially. For example, TEMP_EXTRACT_NAME3 has no effect unless both the options TEMP_EXTRACT_NAME1 and TEMP_EXTRACT_NAME2 are already set.

The most important of these options is TEMP_EXTRACT_NAME1. If TEMP_EXTRACT_NAME1 is set to its default setting (the empty string ""), extraction is disabled and no output is redirected. To enable extraction, set TEMP_EXTRACT_NAME1 to a path name. Extract starts extracting into a file with that name. Choose a path name to a file that is not otherwise in use.

Tip: Set the TEMP_EXTRACT_NAME1 option as TEMPORARY.

You can also use TEMP_EXTRACT_NAME1 to specify the name of the output file, when the TEMP_EXTRACT_APPEND option is set ON. In this case, before you execute the **SELECT** statement, set **WRITE** permission for the user name used to start SAP Sybase IQ (for example, sybase) on the directory or folder containing the named file and on the named file. In append mode, the data extraction facility adds extracted rows to the end of the file and does not overwrite the data that is already in the file. If the output file does not already exist, the data extraction facility creates the file.

Warning! If you choose the path name of an existing file and the TEMP_EXTRACT_APPEND option is set OFF (the default), the file contents are overwritten. This might be what you require if the file is for a weekly report, for example, but not if the file is one of your database files.

The options TEMP_EXTRACT_NAME2 through TEMP_EXTRACT_NAME8 can be used in addition to TEMP_EXTRACT_NAME1 to specify the names of multiple output files.

If you are extracting to a single disk file or a single named pipe, leave the options TEMP_EXTRACT_NAME2 through TEMP_EXTRACT_NAME8 and TEMP_EXTRACT_SIZE1 through TEMP_EXTRACT_SIZE8 at their default values.

When TEMP_EXTRACT_NAME1 is set, you cannot perform these operations:

- **LOAD, DELETE, INSERT, or INSERT...LOCATION** to a table that is the top table in a join
- **INSERT...SELECT**

Also note these restrictions on the data extraction facility:

- Extract works only with data stored in the IQ store.

- Extract does not work on system tables or cross database joins.
- Extract does not work with queries that use user-defined functions or system functions, except for the system functions **suser_id()** and **suser_name()**.
- If you run Interactive SQL with the **-q** (quiet mode) option and the data extraction commands are in a command file, you must first set and make permanent the Interactive SQL option “Show multiple result sets.” If this option is not set, the output file is not created.

To set the “Show multiple result sets” option, select Tools > Options in the Interactive SQL window, then check the box “Show multiple result sets” and click “Make permanent.”

The directory path specified using the `TEMP_EXTRACT_NAME`n options can be overridden with the `TEMP_EXTRACT_DIRECTORY` option.

See also

- *TEMP_EXTRACT_APPEND* Option on page 639
- *TEMP_EXTRACT_DIRECTORY* Option on page 642
- *TEMP_EXTRACT_SIZE*n Options on page 650

TEMP_EXTRACT_NULL_AS_EMPTY Option

Controls the representation of null values in the output of the data extraction facility for an ASCII extraction.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

`TEMP_EXTRACT_NULL_AS_EMPTY` controls the representation of null values in the output of the data extraction facility for ASCII extractions. When the `TEMP_EXTRACT_NULL_AS_EMPTY` option is set to ON, a null value is represented as " (the empty string) for all data types.

The quotes shown above are not present in the extract output file. When the `TEMP_EXTRACT_NULL_AS_EMPTY` option is set to `OFF`, the string 'NULL' is used in all cases to represent a NULL value. `OFF` is the default value.

TEMP_EXTRACT_NULL_AS_ZERO Option

Controls the representation of null values in the output of the data extraction facility for an ASCII extraction.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY PUBLIC OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

`TEMP_EXTRACT_NULL_AS_ZERO` controls the representation of null values in the output of the data extraction facility for ASCII extractions. When `TEMP_EXTRACT_NULL_AS_ZERO` is set to `ON`, a null value is represented as follows:

- '0' for arithmetic type
- " (the empty string) for the `CHAR` and `VARCHAR` character types
- " (the empty string) for dates
- " (the empty string) for times
- " (the empty string) for timestamps

The quotes shown above are not present in the extract output file. When the `TEMP_EXTRACT_NULL_AS_ZERO` option is set to `OFF`, the string 'NULL' is used in all cases to represent a NULL value. `OFF` is the default value.

Note: In SAP Sybase IQ 12.5, an ASCII extract from a `CHAR` or `VARCHAR` column in a table always returns at least four characters to the output file. This is required if `TEMP_EXTRACT_NULL_AS_ZERO` is set to `OFF`, because SAP Sybase IQ needs to write out the word `NULL` for any row in a column that has a null value. Reserving four spaces is not required if `TEMP_EXTRACT_NULL_AS_ZERO` is set to `ON`.

In SAP Sybase IQ 12.6, if `TEMP_EXTRACT_NULL_AS_ZERO` is set to ON, the number of characters that an ASCII extract writes to a file for a CHAR or VARCHAR column equals the number of characters in the column, even if that number is less than four.

TEMP_EXTRACT_QUOTE Option

Specifies the string to be used as the quote to enclose fields in the output of the data extraction facility for an ASCII extraction, when either the `TEMP_EXTRACT_QUOTES` option or the `TEMP_EXTRACT_QUOTES_ALL` option is set ON.

Allowed Values

String

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option specifies the string to be used as the quote to enclose fields in the output of the data extraction facility for an ASCII extraction, if the default value is not suitable. `TEMP_EXTRACT_QUOTE` is used with the `TEMP_EXTRACT_QUOTES` and `TEMP_EXTRACT_QUOTES_ALL` options. The quote string specified in the `TEMP_EXTRACT_QUOTE` option has the same restrictions as the row and column delimiters. The default for this option is the empty string, which SAP Sybase IQ converts to the single quote mark.

The string specified in the `TEMP_EXTRACT_QUOTE` option must occupy from 1 to a maximum of 4 bytes and must be valid in the collation order you are using, if you are using a multibyte collation order. Be sure to choose a string that does not occur in any of the data output strings themselves.

See also

- *TEMP_EXTRACT_COLUMN_DELIMITER Option* on page 641
- *TEMP_EXTRACT_QUOTES Option* on page 648
- *TEMP_EXTRACT_QUOTES_ALL Option* on page 648
- *TEMP_EXTRACT_ROW_DELIMITER Option* on page 649

TEMP_EXTRACT_QUOTES Option

Specifies that string fields are enclosed in quotes in the output of the data extraction facility for an ASCII extraction.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

This option specifies that string fields are enclosed in quotes in the output of the data extraction facility for an ASCII extraction. The string used as the quote is specified in the TEMP_EXTRACT_QUOTE option, if the default is not suitable.

See also

- *TEMP_EXTRACT_COLUMN_DELIMITER Option* on page 641
- *TEMP_EXTRACT_QUOTES_ALL Option* on page 648
- *TEMP_EXTRACT_ROW_DELIMITER Option* on page 649

TEMP_EXTRACT_QUOTES_ALL Option

Specifies that all fields are enclosed in quotes in the output of the data extraction facility for an ASCII extraction.

Allowed Values

ON, OFF

Default

OFF

Scope

Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role.

Description

TEMP_EXTRACT_QUOTES_ALL specifies that all fields are enclosed in quotes in the output of the data extraction facility for an ASCII extraction. The string used as the quote is specified in TEMP_EXTRACT_QUOTE, if the default is not suitable.

See also

- *TEMP_EXTRACT_COLUMN_DELIMITER Option* on page 641
- *TEMP_EXTRACT_QUOTES Option* on page 648
- *TEMP_EXTRACT_QUOTES_ALL Option* on page 648
- *TEMP_EXTRACT_ROW_DELIMITER Option* on page 649

TEMP_EXTRACT_ROW_DELIMITER Option

Specifies the delimiter between rows in the output of the data extraction facility for an ASCII extraction.

Allowed Values

String

Default

" (the empty string)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

TEMP_EXTRACT_ROW_DELIMITER specifies the delimiter between rows in the output of the data extraction facility. In the case of an ASCII extraction, the default is to end the row with a newline on UNIX platforms and with a carriage return/newline pair on Windows platforms.

The delimiter must occupy 1 – 4 bytes and must be valid in the collation order you are using, if you are using a multibyte collation order. Choose a delimiter that does not occur in any of the data output strings. The default for the TEMP_EXTRACT_ROW_DELIMITER option is the empty string. SAP Sybase IQ converts the empty string default for this option to the newline on UNIX platforms and to the carriage return/newline pair on Windows platforms.

See also

- *TEMP_EXTRACT_COLUMN_DELIMITER* Option on page 641
- *TEMP_EXTRACT_QUOTES* Option on page 648
- *TEMP_EXTRACT_QUOTES_ALL* Option on page 648

TEMP_EXTRACT_SIZE_n Options

Specifies the maximum sizes of the corresponding output files used by the data extraction facility.

Allowed Values

There are eight options: TEMP_EXTRACT_SIZE1 through TEMP_EXTRACT_SIZE8.

Device Type	Size
Disk file	AIX and HP-UX: 0 – 64GB Sun Solaris & Linux: 0 – 512GB Windows: 0 – 128GB
Tape*	524288KB (0.5GB)
Other	9007199254740992KB (8192 Petabytes “unlimited”)

*Tape devices currently are not supported.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

TEMP_EXTRACT_SIZE1 through TEMP_EXTRACT_SIZE8 are used to specify the maximum sizes of the corresponding output files used by the data extraction facility. TEMP_EXTRACT_SIZE1 specifies the maximum size of the output file specified by TEMP_EXTRACT_NAME1, TEMP_EXTRACT_SIZE2 specifies the maximum size of the output file specified by TEMP_EXTRACT_NAME2, and so on.

When large file systems, such as JFS2, support file size larger than the default value, set `TEMP_EXTRACT_SIZE n` to the value that the file system allows. For example, to support ITB set option:

```
TEMP_EXTRACT_SIZE1 = 1073741824 KB
```

If you are extracting to a single disk file or a single named pipe, leave the options `TEMP_EXTRACT_NAME2` through `TEMP_EXTRACT_NAME8` and `TEMP_EXTRACT_SIZE1` through `TEMP_EXTRACT_SIZE8` at their default values.

The `TEMP_EXTRACT_SIZE n` options are not compatible with `TEMP_EXTRACT_APPEND`. If you try to restrict the size of the extract append output file, SAP Sybase IQ reports an error.

See also

- *TEMP_EXTRACT_NAME n Options* on page 643

TEMP_EXTRACT_SWAP Option

In combination with the `TEMP_EXTRACT_BINARY` option, specifies the type of extraction performed by the data extraction facility.

Allowed values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Use this option with the `TEMP_EXTRACT_BINARY` option to specify the type of extraction performed by the data extraction facility.

Table 21. Extraction Option Settings for Extraction Type

Extraction type	TEMP_EXTRACT_BINARY	TEMP_EXTRACT_SWAP
binary	ON	OFF

Extraction type	TEMP_EXTRACT_BINARY	TEMP_EXTRACT_SWAP
binary/swap	ON	ON
ASCII	OFF	OFF

The default extraction type is ASCII.

See also

- *TEMP_EXTRACT_BINARY Option* on page 640

TEMP_RESERVED_DBSPACE_MB Option

Controls the amount of space SAP Sybase IQ reserves in the temporary IQ store.

Allowed Values

Integer greater than or equal to 200 in megabytes

Default

200; SAP Sybase IQ actually reserves a maximum of 50% and a minimum of 1% of the last read-write file in IQ_SYSTEM_TEMP

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately. The server does not need to be restarted in order to change reserved space size.

Description

TEMP_RESERVED_DBSPACE_MB lets you control the amount of space SAP Sybase IQ sets aside in your temporary IQ store for certain small but critical data structures used during release savepoint, commit, and checkpoint operations. For a production database, set this value between 200MB and 1GB. The larger your IQ page size and number of concurrent connections, the more reserved space you need.

Reserved space size is calculated as a maximum of 50% and a minimum of 1% of the last read-write file in IQ_SYSTEM_TEMP.

TEMP_SPACE_LIMIT_CHECK Option

Checks for catalog store temporary space on a per connection basis.

Allowed Values

ON, OFF (no limit checking occurs)

Default

ON

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

When TEMP_SPACE_LIMIT_CHECK is ON, the database server checks the amount of catalog store temporary file space that a connection uses. If a connection requests more than its quota of temporary file space when this option is set to OFF, a fatal error can occur. When this option is set to ON, if a connection requests more than its quota of temporary file space, the request fails and the error “Temporary space limit exceeded” is returned.

Two factors are used to determine the temporary file quota for a connection: the maximum size of the temporary file, and the number of active database connections. The maximum size of the temporary file is the sum of the current size of the file and the amount of disk space available on the partition containing the file. When limit checking is turned on, the server checks a connection for exceeding its quota when the temporary file has grown to 80% or more of its maximum size, and the connection requests more temporary file space. Once this happens, any connection fails that uses more than the maximum temporary file space divided by the number of active connections.

Note: This option is unrelated to IQ temporary store space. To constrain the growth of IQ temporary space, use the QUERY_TEMP_SPACE_LIMIT option and MAX_TEMP_SPACE_PER_CONNECTION option.

You can obtain information about the space available for the temporary file using the **sa_disk_free_space** system procedure.

Example

A database is started with the temporary file on a drive with 100MB free and no other active files on the same drive. The available temporary file space is 100MB. The DBA enters:

```
SET OPTION PUBLIC.TEMP_SPACE_LIMIT_CHECK = 'ON'
```

As long as the temporary file stays below 80MB, the server behaves as it did before. Once the file reaches 80MB, the new behavior might occur. Assume that with 10 queries running, the temporary file needs to grow. When the server finds that one query is using more than 8MB of temporary file space, that query fails.

TEXT_DELETE_METHOD Option

Specifies the algorithm used during a delete in a **TEXT** index.

Allowed Values

0 – 2

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

Users must be licensed for the Unstructured Data Analytics Option to use **TEXT** indexes.

TIME_FORMAT Option

Sets the format used for times retrieved from the database.

Allowed values

A string composed of the symbols HH, NN, MM, SS, separated by colons.

Default

'HH:NN:SS.SSS'

For Open Client and JDBC connections the default is also set to HH:NN:SS.SSS.

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The format is a string using these symbols:

- hh – Two-digit hours (24 hour clock).
- nn – Two-digit minutes.
- mm – Two-digit minutes if following a colon (as in 'hh:mm').
- ss[s...s] – Two-digit seconds plus optional fraction.

Each symbol is substituted with the appropriate data for the date being formatted. Any format symbol that represents character rather than digit output can be in uppercase, which causes the substituted characters also to be in uppercase. For numbers, using mixed case in the format string suppresses leading zeros.

Multibyte characters are not supported in format strings. Only single-byte characters are allowed, even when the collation order of the database is a multibyte collation order like 932JPN.

See also

- *DATE_FORMAT Option* on page 512
- *RETURN_DATE_TIME_AS_STRING Option* on page 619

TIMESTAMP_FORMAT Option

Sets the format used for timestamps retrieved from the database.

Allowed Values

A string composed of the symbols listed below.

Default

'YYYY-MM-DD HH:NN:SS.SSS'

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

The format is a string using these symbols:

Table 22. TIMESTAMP_FORMAT String Symbols

Symbol	Description
yy	2-digit year.
yyyy	4-digit year.
mm	2-digit month, or two digit minutes if following a colon (as in 'hh:mm').
mmm	3-character short form for name of the month of year

Symbol	Description
mmmm[m...]	Character long form for month name—as many characters as there are m's, until the number of m's specified exceeds the number of characters in the month's name.
dd	2-digit day of month.
ddd	3-character short form for name of the day of week.
dddd[d...]	Character long form for day name—as many characters as there are d's, until the number of d's specified exceeds the number of characters in the day's name.
hh	2-digit hours.
nn	2-digit minutes.
ss.SSS	Seconds (ss) and fractions of a second (SSS), up to six decimal places. Not all platforms support timestamps to a precision of six places.
aa	a.m. or p.m. (12-hour clock).
pp	p.m. if needed (12-hour clock.)

Each symbol is substituted with the appropriate data for the date being formatted. Any format symbol that represents character rather than digit output can be in uppercase, which causes the substituted characters also to be in uppercase. For numbers, using mixed case in the format string suppresses leading zeros.

Multibyte characters are not supported in format strings. Only single-byte characters are allowed, even when the collation order of the database is a multibyte collation order like 932JPN.

See also

- *DATE_FORMAT Option* on page 512
- *RETURN_DATE_TIME_AS_STRING Option* on page 619

TOP_NSORT_CUTOFF_PAGES Option

Sets the result size threshold for **TOP N** algorithm selection.

Allowed Values

1 – 1000

Default

1

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at

the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

TOP_NSORT_CUTOFF_PAGES sets the threshold, measured in pages, where evaluation of a query that contains both a **TOP** clause and **ORDER BY** clause switches algorithms from ordered list-based processing to sort-based processing. Ordered list processing performs better in cases where the **TOP N** value is smaller than the number of result rows. Sort-based processing performs better for large **TOP N** values.

In some cases, increasing TOP_NSORT_CUTOFF_PAGES can improve performance by avoiding sort-based processing.

See also

- *SELECT Statement* on page 417

TRIM_PARTIAL_MBC Option

Allows automatic trimming of partial multibyte character data.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. Takes effect immediately.

Description

Provides consistent loading of data for collations that contain both single-byte and multibyte characters. When TRIM_PARTIAL_MBC is ON:

- A partial multibyte character is replaced with a blank when loading into a CHAR column.
- A partial multibyte character is truncated when loading into a VARCHAR column.

When TRIM_PARTIAL_MBC is OFF, normal CONVERSION_ERROR semantics are in effect.

See also

- *CONVERSION_ERROR Option [TSQL]* on page 499

TRUSTED_CERTIFICATES_FILE Option

Specifies the trust relationship for outbound Transport Layer Security (TLS) connections made by LDAP User Authentication, INC, and MIPC connections.

Allowed Values

A valid network path to the location of a TXT file containing the list of trusted certificate authorities that sign server certificates.

Default

NULL, meaning that no outbound TLS connection can be started because there are no trusted certificate authorities.

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Takes effect immediately.

Description

This option identifies the path to the location of the list of trusted certificate authorities. The list must be stored in a TXT file. The file may be shared in a location in a Windows environment on the local drive to be used by all SAP Sybase applications on that machine.

TSQL_VARIABLES Option [TSQL]

Controls whether the @ sign can be used as a prefix for Embedded SQL host variable names.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When `TSQL_VARIABLES` is set to `ON`, you can use the `@` sign instead of the colon as a prefix for host variable names in Embedded SQL. This is implemented primarily for the Open Server Gateway.

USER_RESOURCE_RESERVATION Option

Adjusts memory use for the number of current users.

Allowed Values

Integer

Default

1

Scope

Option can be set at the database (`PUBLIC`) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the `PUBLIC` value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the `SET ANY PUBLIC OPTION` system privilege to set this option. Can be set temporary for an individual connection or for the `PUBLIC` role. Takes effect immediately.

Description

SAP Sybase IQ tracks the number of open cursors and allocates memory accordingly. In certain circumstances, you can use this option to adjust the minimum number of current cursors that SAP Sybase IQ thinks is currently using the product, and allocate memory from the temporary cache more sparingly.

Set this option only after careful analysis shows it is actually required. If you need to set this parameter, contact Sybase Technical Support with details.

VERIFY_PASSWORD_FUNCTION Option

Specifies a user-supplied authentication function that can be used to implement password rules.

Allowed Values

String

Default

" (the empty string). (No function is called when a password is set.)

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY SECURITY OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

When the VERIFY_PASSWORD_FUNCTION option value is set to a valid string, the statement **GRANT CONNECT TO** *userid* **IDENTIFIED BY** *password* calls the function specified by the option value.

The option value requires the form *owner.function_name* to prevent users from overriding the function.

The function takes two parameters:

- *user_name* VARCHAR(128)
- *new_pwd* VARCHAR(255)

The return value type is VARCHAR(255).

If VERIFY_PASSWORD_FUNCTION is set, you cannot specify more than one userid and password with the **GRANT CONNECT** statement.

Example

The following sample code defines a table and a function and sets some login policy options. Together they implement advanced password rules that include requiring certain types of characters in the password, disallowing password reuse, and expiring passwords. The function is called by the database server with the verify_password_function option when a user ID is created or a password is changed. The application can call the procedure specified by the post_login_procedure option to report that the password should be changed before it expires.

```
-- only DBA should have privileges on this table
CREATE TABLE DBA.t_pwd_history(
    pk          INT          DEFAULT AUTOINCREMENT PRIMARY KEY,
    user_name   CHAR(128),   -- the user whose password is set
    pwd_hash    CHAR(32) );  -- hash of password value to detect
                             -- duplicate passwords

-- called whenever a non-NULL password is set
-- to verify the password conforms to password rules
CREATE FUNCTION DBA.f_verify_pwd( uid      VARCHAR(128),
                                  new_pwd  VARCHAR(255) )
RETURNS VARCHAR(255)
BEGIN
    -- enforce password rules
```

```

-- enforce minimum length (can also be done with
-- min_password_length option)
IF length( new_pwd ) < 6 THEN
    RETURN 'password must be at least 6 characters long';
END IF;

-- number of lowercase characters IN new_pwd
SELECT count(*) INTO num_lower_chars
    FROM pwd_chars WHERE CAST( c AS BINARY ) BETWEEN 'a' AND 'z';

-- enforce rules based on characters contained in new_pwd
IF ( SELECT count(*) FROM pwd_chars WHERE c BETWEEN '0' AND '9' )
    < 1 THEN
    RETURN 'password must contain at least one numeric digit';
ELSEIF length( pwd_alpha_only ) < 2 THEN
    RETURN 'password must contain at least two letters';
ELSEIF num_lower_chars = 0
    OR length( pwd_alpha_only ) - num_lower_chars = 0 THEN
    RETURN 'password must contain both upper- and lowercase
characters';
END IF;

-- not the same as any user name
-- (this could be modified to check against a disallowed words
table)
IF EXISTS( SELECT * FROM SYS.SYSUSER
            WHERE lower( user_name ) IN
( lower( pwd_alpha_only ),
                                lower( new_pwd ) ) ) THEN
    RETURN 'password or only alphabetic characters in password '
||
    'must not match any user name';
END IF;

-- not the same as any previous password for this user
IF EXISTS( SELECT * FROM t_pwd_history
            WHERE user_name = uid
            AND pwd_hash = hash( uid || new_pwd, 'md5' ) ) THEN
    RETURN 'previous passwords cannot be reused';
END IF;

-- save the new password
INSERT INTO t_pwd_history( user_name, pwd_hash )
    VALUES( uid, hash( uid || new_pwd, 'md5' ) );

RETURN( NULL );
END;

ALTER FUNCTION DBA.f_verify_pwd SET HIDDEN;
GRANT EXECUTE ON DBA.f_verify_pwd TO PUBLIC;
SET OPTION PUBLIC.verify_password_function = 'DBA.f_verify_pwd';

-- All passwords expire in 180 days. Expired passwords can be changed
-- by the user using the NewPassword connection parameter.
ALTER LOGIN POLICY DEFAULT password_life_time = 180;

```

```
-- If an application calls the procedure specified by the
-- post_login_procedure option, then the procedure can be used to
-- warn the user that their password is about to expire. In
particular,
-- Interactive SQL calls the post_login_procedure.
ALTER LOGIN POLICY DEFAULT password_grace_time = 30;
```

To turn the option off, set it to the empty string:

```
SET OPTION PUBLIC.VERIFY_PASSWORD_FUNCTION = ''
```

See also

- *ALTER FUNCTION Statement* on page 16
- *GRANT CONNECT Statement* on page 306

WASH_AREA_BUFFERS_PERCENT Option

Specifies the percentage of the buffer caches above the wash marker.

Allowed Values

1 – 100

Default

20

Scope

Option can be set at the database (PUBLIC) level only.

Requires the SET ANY SYSTEM OPTION system privilege to set this option. You must shut down the database and restart it for the change to take effect.

Description

SAP Sybase IQ buffer caches are organized as a long MRU/LRU chain. The area above the wash marker is used to sweep out (that is, write) dirty pages to disk.

In the IQ Monitor **-cache** report, the Gdirty column shows the number of times the LRU buffer was grabbed in a “dirty” (modified) state. If GDirty is greater than 0 for more than a brief time, you might need to increase SWEEPER_THREADS_PERCENT or WASH_AREA_BUFFERS_PERCENT.

Note: Before changing this option, check the value of the CACHE_AFFINITY_PERCENT option. WASH_AREA_BUFFERS_PERCENT affects the LRU side of the buffer cache and CACHE_AFFINITY_PERCENT affects the MRU side. The total of these two values cannot exceed 100 percent.

The default setting of this option is almost always appropriate. Occasionally, SAP Sybase Technical Support might ask you to increase this value.

See also

- *SWEeper_THREADS_Percent* Option on page 638

WAIT_FOR_COMMIT Option

Determines when foreign key integrity is checked as data is manipulated.

Allowed Values

ON, OFF

Default

OFF

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

If this option is set to ON, the database does not check foreign key integrity until the next **COMMIT** statement. Otherwise, all foreign keys not created with the `CHECK ON COMMIT` option are checked as they are inserted, updated, or deleted.

WD_DELETE_METHOD Option

Specifies the algorithm used during a delete in a **WD** index.

Allowed Values

Value	Action
0	The delete method is selected by the cost model. Cost model only selects either mid or large method for deletion.
1	Forces small method for deletion. Small method is useful when the number of rows being deleted is a very small percentage of the total number of rows in the table. Small delete can randomly access the index, causing cache thrashing with large datasets.

Value	Action
2	Forces large method for deletion. This algorithm scans the entire index searching for rows to delete. Large method is useful when the number of rows being deleted is a high percentage of the total number of rows in the table.
3	Forces mid method for deletion. Mid method is a variation of the small method that accesses the index in order and is generally faster than the small method.

Default

0

Scope

Option can be set at the database (PUBLIC) or user level. When set at the database level, the value becomes the default for any new user, but has no impact on existing users. When set at the user level, overrides the PUBLIC value for that user only. No system privilege is required to set option for self. System privilege is required to set at database level or at user level for any user other than self.

Requires the SET ANY PUBLIC OPTION system privilege to set this option. Can be set temporary for an individual connection or for the PUBLIC role. Takes effect immediately.

Description

WD_DELETE_METHOD specifies the algorithm used during a delete operation in a **WD** index. When this option is not set or is set to 0, the delete method is selected by the cost model. The cost model considers the CPU related costs as well as I/O related costs in selecting the appropriate delete algorithm. The cost model takes into account:

- Rows deleted
- Index size
- Width of index data type
- Cardinality of index data
- Available temporary cache
- Machine related I/O and CPU characteristics
- Available CPUs and threads

Example

Force the large method for deletion from a **WD** index:

```
SET TEMPORARY OPTION WD_DELETE_METHOD = 2
```

Index

A

- AES encryption algorithm
 - CREATE DATABASE statement 107
- AGGREGATION_PREFERENCE option 478
- aliases 418
 - in SELECT statement 425
 - in the DELETE statement 248
- ALL keyword in SELECT statement 417
- ALLOCATE DESCRIPTOR statement
 - syntax 5
- ALLOW_NULLS_BY_DEFAULT option 479
- ALLOW_READ_CLIENT_FILE option 485
- ALLOW_SNAPSHOT_VERSIONING option 480
- ALTER DATABASE statement
 - syntax 7
- ALTER DATABASE UPGRADE statement 7
- ALTER DBSPACE statement
 - syntax 9
- ALTER DOMAIN statement
 - syntax 13
- ALTER EVENT statement
 - syntax 14
- ALTER FUNCTION statement
 - syntax 16
- ALTER INDEX statement
 - errors 17
- ALTER LDAP SERVER statement 20
- ALTER LOGICAL SERVER statement
 - syntax 22
- ALTER LOGIN POLICY statement
 - syntax 24
- ALTER LS POLICY statement
 - syntax 32
- ALTER MULTIPLEX RENAME statement 34
- ALTER MULTIPLEX SERVER statement 35
- ALTER PROCEDURE statement
 - syntax 36
- ALTER ROLE statement 38
- ALTER SERVER statement
 - syntax 41
- ALTER SERVICE statement
 - syntax 44
- ALTER TABLE statement
 - syntax 52
- ALTER TEXT CONFIGURATION
 - syntax 68
- ALTER TEXT INDEX
 - syntax 66
- ALTER USER statement 71
- ALTER VIEW statement
 - RECOMPILE 52
 - syntax 75, 77
- altering
 - databases 7
 - functions 16
 - text configuration object 68
 - TEXT index 66
- ANSI_CLOSE_CURSORS_AT_ROLLBACK
 - option 480
- ANSI_PERMISSIONS option 481
- ANSI_SUBSTRING option 483
- ANSI_UPDATE_CONSTRAINTS option 484
- ANSINULL option 482
- archive backup
 - restoring 394
- archive devices
 - maximum for parallel backup 78
- ASE_BINARY_DISPLAY
 - database option 486
- ASE_FUNCTION_BEHAVIOR
 - database option 487
 - with HEXTOINT 487
 - with INTTOHEX 487
- AT clause
 - CREATE EXISTING TABLE 129
- AUDITING option 488
- autoincrement
 - primary key values 276
- AUTOINCREMENT column default 205

B

- B-tree pages 491
- BACKUP statement
 - number of archive devices 78
 - syntax 78
- backups
 - speed 78
 - verifying 389
- BASE_TABLES_IN_RLV option 488

Index

- BEGIN DECLARE SECTION statement
 - syntax 237
- BEGIN PARALLEL IQ statement 87, 223
- BEGIN TRANSACTION statement
 - Transact-SQL 88
- BEGIN... END statement
 - syntax 84
- binary data
 - controlling implicit conversion 499
- bind variables
 - DESCRIBE statement 251
 - EXECUTE statement 276
 - OPEN statement 369
- blanks
 - trimming trailing 343
- block fetches
 - FETCH statement 283
- BLOCKING option 490, 492
- BLOCKING_TIMEOUT option 490
- BREAK statement
 - Transact-SQL 460
- BT_PREFETCH_MAX_MISS option 491
- BTREE_PAGE_SPLIT_PAD_PERCENT option 493
- buffer cache
 - partitioning 494
- buffers
 - disabling operating system buffering 599, 600
- bulk load 343
- BYE statement
 - syntax 282
- C**
- CACHE_PARTITIONS option 494
- CALL statement
 - syntax 90
 - Transact-SQL 278
- CASE statement
 - syntax 92
- catalog store 302, 417
- catalog temporary files
 - preventing connections from exceeding quota 652
- CHAINED option 495
- change password
 - grant 304
 - revoking 398
- character sets
 - client file bulk load 343
 - errors on conversions 596
- CHECK conditions
 - about 205
- CHECK ON COMMIT clause
 - referential integrity 205
- CHECKPOINT statement
 - syntax 94
- CHECKPOINT_TIME option 496
- CIS
 - remote data access 497
- CIS_ROWSET_SIZE option
 - about 497
- classes
 - installing 336
 - removing 387
- CLEAR statement
 - syntax 94
- client file bulk load
 - character sets 343
 - errors 343
 - rollback 343
- CLOSE statement
 - syntax 95
- CLOSE_ON_ENDTRANS option 497
- code pages
 - DEFAULT_ISQL_ENCODING option 520
- collation
 - SORT_COLLATION option 628
- collations
 - client file bulk load 343
- columns 418
 - altering 52
 - constraints 205
 - naming 3
- command files
 - parameters 374
- COMMENT statement
 - syntax 96
- COMMIT statement
 - syntax 102
- COMMIT TRANSACTION statement
 - Transact-SQL 102
- compatibility options
 - ASE_FUNCTION_BEHAVIOR 487
 - CONTINUE_AFTER_RAISERROR 498
 - CONVERSION_ERROR 499
 - ON_TSQL_ERROR 598
- compound statements
 - about 84

- concurrency
 - locking tables 362
- CONFIGURE statement
 - syntax 104
- CONNECT privilege
 - GRANT statement 306
- CONNECT statement
 - revoke 399
 - syntax 104
- connection_property function
 - about 464
- connections
 - dbisql 254
 - DEDICATED_TASK option 517
 - establishing 24
 - logging 569
 - logical servers 30
- console
 - displaying messages on 366
- contains-expression
 - FROM clause 295
- CONTINUE statement
 - Transact-SQL 460
- CONTINUE_AFTER_RAISE_ERROR option
 - 498
- control statements
 - CALL statement 90
 - CASE statement 92
 - IF statement 323
 - LEAVE statement 342
 - LOOP statement 364
 - Transact-SQL GOTO statement 304
 - Transact-SQL IF statement 325
 - Transact-SQL WHILE statement 460
- CONVERSION_ERROR option 499
- CONVERSION_MODE option 499
- CONVERT_VARCHAR_TO_1242 option 506
- COOPERATIVE_COMMIT_TIMEOUT option
 - 506
- COOPERATIVE_COMMITS option 507
- correlation names
 - in the DELETE statement 248
- CREATE DATABASE statement
 - syntax 107
- CREATE DBSPACE statement
 - syntax 117
- CREATE DOMAIN statement
 - syntax 121
- CREATE EVENT statement
 - syntax 123
- CREATE EXISTING TABLE statement
 - proxy tables 129
- CREATE EXTERNLOGIN statement
 - INSERT...LOCATION 332
 - syntax 131
- CREATE FUNCTION statement
 - external environment 139
 - Java 139
 - syntax 133
 - UDF 139
- CREATE INDEX statement 87
 - syntax 142
 - table use 146
- CREATE LDAP SERVER statement 150
- CREATE LOGICAL SERVER statement 153
- CREATE LOGIN POLICY statement
 - syntax 155
- CREATE LS POLICY statement
 - syntax 162
- CREATE MESSAGE statement
 - Transact-SQL 163
- CREATE MULTIPLEX SERVER statement 164
- CREATE ON statement
 - revoke 400
- CREATE PROCEDURE statement
 - syntax 165
 - Transact-SQL 172
- CREATE PROCEDURE statement for external
 - procedures
 - syntax 175, 181, 183
- CREATE ROLE statement 187
- CREATE SCHEMA statement
 - syntax 189
- CREATE SERVER statement
 - INSERT...LOCATION 332
 - syntax 192
- CREATE SERVICE statement
 - syntax 194
- CREATE statement
 - grant 308
- CREATE TABLE statement
 - syntax 205
- CREATE TEXT CONFIGURATION
 - syntax 222
- CREATE TEXT INDEX
 - syntax 223
- CREATE USER statement 230

Index

- CREATE VARIABLE statement
 - syntax 232
- CREATE VIEW statement
 - syntax 234
- CREATE_HG_AND_FORCE_PHYSICAL_DELETE 509
- CREATE_HG_WITH_EXACT_DISTINCTS 508
- creating
 - data types 121
 - external stored procedures 175, 181, 183
 - proxy tables 129
 - stored procedures 165
 - text configuration object 222
 - TEXT index 223
- creating as a group 87
- creator 3
- CUBE operator 421
- CURSOR_WINDOW_ROWS option 510
- cursors
 - closing 95
 - database options 465
 - declaring 240, 245
 - deleting rows from 250
 - DESCRIBE 251
 - fetching 283
 - FOR UPDATE clause 240
 - INSENSITIVE 240
 - inserting rows using 379
 - looping over 286
 - OPEN statement 369
 - sensitivity 243
 - WITH HOLD clause 369
- D**
- data
 - exporting from tables into files 371
- data type conversion
 - CONVERSION_MODE option 499
 - errors 499
- data types
 - altering user-defined 13
 - creating 121
 - dropping user-defined 255
 - performance for joins 302
- database files
 - altering 9
 - creating 117
- database option
 - ENABLE_LOB_VARIABLES 530
- database options
 - cursors 465
 - DEBUG_MESSAGES option 516
 - DEDICATED_TASK 517
 - duration 465
 - FLATTEN_SUBQUERIES 636
 - FORCE_DROP 532
 - FP_LOOKUP_SIZE_PPM 535
 - initial settings 468
 - maximum string length 431, 464
 - ODBC_DISTINGUISH_CHAR_AND_VARCHAR 595
 - ON_CHARSET_CONVERSION_FAILURE 596
 - POST_LOGIN_PROCEDURE 601
 - PRESERVE_SOURCE_FORMAT 606
 - PUBLIC 466
 - RETURN_DATE_TIME_AS_STRING 619
 - ROUND_TO_EVEN 620
 - SECURITY 466
 - SUBQUERY_FLATTENING_PERCENT 634
 - SUBQUERY_FLATTENING_PREFERENCE 635
 - SUPPRESS_TDS_DEBUGGING 637
 - SYSTEM 467
 - TDS_EMPTY_STRING_IS_NULL 638
- database servers
 - starting 439
 - stopping 442
- databases
 - altering 7
 - creating 107
 - deleting files 260
 - disabling jConnect support 7
 - enabling jConnect support 7
 - loading data into 343
 - starting 438
 - stopping 441
 - upgrading 7
- DATE_FIRST_DAY_OF_WEEK option 510
- DATE_FORMAT option 512
- DATE_ORDER option 514
- DBCC_LOG_PROGRESS
 - database option 514
- DBCC_PINNABLE_CACHE_PERCENT
 - database option 515
- dbisql
 - connecting to a database 104

- options 434
- dbo user ID
 - views owned by 255
- dbspaces
 - altering 9
 - creating 117
 - dropping 255
 - setting offline 9
 - virtual backup 78
- DEALLOCATE DESCRIPTOR
 - syntax 237
- DEBUG_MESSAGES option
 - description 516
- debugging
 - controlling MESSAGE statement behavior 366
 - DEBUG_MESSAGES option 516
- declaration section 237
- DECLARE CURSOR statement
 - syntax 240
 - Transact-SQL syntax 245
- DECLARE LOCAL TEMPORARY TABLE
 - statement
 - syntax 246
- DECLARE statement
 - syntax 84, 238
- DECLARE TEMPORARY TABLE statement
 - syntax 246
- DEDICATED_TASK option
 - description 517
- DEFAULT_DBSPACE option 517
- DEFAULT_DISK_STRIPING option 519
- DEFAULT_HAVING_SELECTIVITY_PPM
 - option 519
- DEFAULT_ISQL_ENCODING option
 - description 520
- DEFAULT_KB_PER_STRIPE option 521
- DEFAULT_LIKE_MATCH_SELECTIVITY_PP
 - M option 522
- DEFAULT_LIKE_RANGE_SELECTIVITY_PPM
 - option 523
- DEFAULT_PROXY_TABLE_ROW_COUNT
 - option 523
- DEFAULT_TABLE_UDF_ROW_COUNT option
 - 524
- DELAYED_COMMIT_TIMEOUT option 525
- DELAYED_COMMITS option 525
- DELETE (positioned) statement
 - SQL syntax 250
- DELETE statement
 - syntax 248
- deleting
 - rows from cursors 250
- deleting all rows from a materialized view 444
- deleting all rows from a table 444
- delimiters
 - example 142
- deprecated database options 469
- DESCRIBE statement
 - syntax 251
- descriptor
 - allocating memory 5
 - deallocating 237
 - DESCRIBE statement 251
 - EXECUTE statement 276
 - FETCH statement 283
 - getting 303
 - PREPARE statement 375
- descriptor areas
 - UPDATE (positioned) statement 451
- descriptors
 - setting 430
- direct I/O 599, 600
- DISCONNECT statement
 - syntax 254
- disjunction of 420
- disjunction of subquery predicates 420
- disk space
 - notifying when low 123
- DISK_STRIPING option 526
- displaying
 - messages 366
- DISTINCT keyword in SELECT statement 417
- distributed query processing
 - performance 32, 162
- DIVIDE_BY_ZERO_ERROR option 526
- domains 121
 - altering 13
- DQP
 - performance 32, 162
- DQP_ENABLED option 527
- DQP_ENABLED_OVER_NETWORK option 528
- DROP CONNECTION statement
 - syntax 259
- DROP DATABASE statement
 - syntax 260
- DROP DATATYPE statement
 - syntax 255

Index

- DROP DBSPACE statement
 - syntax 255
 - DROP DOMAIN statement
 - syntax 255
 - DROP EVENT
 - syntax 255
 - DROP EXTERNLOGIN statement
 - syntax 261
 - DROP FUNCTION statement
 - syntax 255
 - DROP INDEX statement
 - syntax 255
 - DROP LDAP SERVER statement 262
 - DROP LOGICAL SERVER statement 264
 - DROP LOGIN POLICY statement
 - syntax 263
 - DROP LS POLICY statement
 - syntax 265
 - DROP MESSAGE
 - syntax 255
 - DROP MULTIPLEX SERVER statement 265
 - DROP PROCEDURE statement
 - syntax 255
 - DROP ROLE statement 266
 - DROP SERVER statement
 - syntax 269
 - DROP SERVICE statement
 - syntax 269
 - DROP statement
 - syntax 255
 - DROP STATEMENT statement
 - syntax 271
 - DROP TABLE
 - IDENTITY_INSERT option 255
 - DROP TABLE statement
 - syntax 255
 - DROP TEXT CONFIGURATION
 - syntax 272
 - DROP TEXT INDEX
 - syntax 273
 - DROP USER statement 275
 - DROP VARIABLE statement
 - syntax 276
 - DROP VIEW statement
 - restriction 255
 - syntax 255
 - dropping
 - text configuration object 272
 - TEXT index 273
 - users 400
 - views 255
 - dropping partitions 52
 - DUMMY 302
 - dummy IQ table 302
 - DYNAMIC SCROLL cursors 240
- ## E
- EARLY_PREDICATE_EXECUTION option 528
 - embedded SQL
 - DELETE (positioned) statement syntax 250
 - PUT statement syntax 379
 - ENABLE_ASYNC_IO option 530
 - ENABLE_LOB_VARIABLES option 530
 - encryption
 - TDS password 333
 - encryption algorithms
 - CREATE DATABASE statement 107
 - END DECLARE STATEMENT
 - syntax 237
 - END keyword 84
 - END PARALLEL IQ
 - CREATE TEXT INDEX 223
 - END PARALLEL IQ statement 87
 - error handling
 - Transact-SQL procedures 598
 - errors
 - during character conversions 596
 - RAISERROR statement 381
 - SIGNAL statement 437
 - Transact-SQL procedures 598
 - escape character
 - OUTPUT SQL statement 371
 - event
 - monitoring disk space 123
 - event handler
 - altering 14
 - creating 123
 - triggering 443
 - events
 - altering 14
 - creating 123
 - dropping 255
 - triggering 443
 - EXCEPTION statement
 - syntax 84
 - EXECUTE IMMEDIATE statement
 - syntax 279

EXECUTE statement
 grant 310
 revoke 401
 syntax 276
 Transact-SQL 278

EXIT statement
 syntax 282

exporting data
 from tables into files 371
 SELECT statement 417

EXTENDED_JOIN_SYNTAX option 531

external procedures
 creating 175, 181, 183

external stored procedures
 creating 175, 181, 183

F

FETCH statement
 syntax 283

files
 dbspaces 9, 117
 exporting data from tables into 371
 setting offline 9
 setting online 9

FIRST 418

FLATTEN_SUBQUERIES option 636

for columns 418

FOR JSON statement
 syntax 289

FOR statement
 syntax 286

FORCE_DROP option 532

FORCE_NO_SCROLL_CURSORS option 533

FORCE_UPDATABLE_CURSORS option 534

foreign keys
 integrity constraints 205
 unnamed 205

FORWARD TO statement
 syntax 294

FP indexes
 cache allocated 535

FP_LOOKUP_SIZE option 535

FP_LOOKUP_SIZE_PPM option 535

FP_NBIT_AUTO_LIMIT 536

FP_NBIT_ENFORCE_LIMITS 538

FP_NBIT_IQ15_COMPATIBILITY 539

FP_NBIT_LOOKUP_MB 540

FP_NBIT_ROLLOVER_MAX_MB 542

FP_PREDICATE_WORKUNIT_PAGES option
 543

FPL_EXPRESSION_MEMORY_KB option 543

FROM clause 302, 417, 419
 contains-expression 295
 selects from stored procedure result sets 425
 syntax 295

functions
 altering 16
 creating 133
 dropping 255
 user-defined 396

G

GARRAY_FILL_FACTOR_PERCENT option
 544

GARRAY_PAGE_SPLIT_PAD_PERCENT option
 545

GARRAY_PREFETCH_SIZE option 545, 546

GET DESCRIPTOR statement
 syntax 303

global transaction
 suspending 589

GOTO statement
 Transact-SQL 304

GRANT CHANGE PASSWORD statement 304

GRANT object-level privileges 309

GRANT ROLE statement 312

GRANT SET USER statement 317

GRANT statement
 CONNECT privilege 306

GRANT system privilege statement 319

GROUP BY clause 420

grouping 87

H

HASH_THRASHING_PERCENT option 548

HEADER SKIP option
 LOAD TABLE statement 343

heading name 418

HG index
 multicolumn with NULL 148
 NULL values 148

HG indexes
 improving query performance 491

HG_DELETE_METHOD option 548

HG_SEARCH_RANGE option 549

host variables
 declaring 237

Index

syntax 3
HTTP_SESSION_TIMEOUT option 550

I

I/O
 direct 599, 600
IDENTITY column
 and DROP TABLE 255
IDENTITY_ENFORCE_UNIQUENESS option 551
IDENTITY_INSERT option
 dropping tables 255
IF statement
 syntax 323
 Transact-SQL 325
impact of FROM clause 302
in SELECT statement 418
IN_SUBQUERY_PREFERENCE option 552
INCLUDE statement
 syntax 326
IDENTITY_INSERT option 551
INDEX_ADVISOR option 554
INDEX_ADVISOR_MAX_ROWS option 556
INDEX_PREFERENCE option 557
indexes 87
 creating 142
 dropping 255
 lookup pages 535
 multicolumn 147
 multicolumn HG and NULL 148
 naming 146
 owner 142
 table use 146
 unique 142
indicator variables 3
INFER_SUBQUERY_PREDICATES option 558
INSERT
 syntax 327
 wide 276
INSERT statement
 WORD SKIP option 327
inserting
 rows using cursors 379
INSTALL JAVA statement
 syntax 336
INTEGRATED LOGIN
 revoke 401
INTEGRATED LOGIN statement
 grant 311

Interactive SQL
 OUTPUT statement syntax 371
 specifying code page for reading and writing to files 520
Interactive SQL options
 DEFAULT_ISQL_ENCODING 520
INTO clause 418
IQ store
 reserving space 572
 reserving temporary space 652
IQ UNIQUE
 alternative method 585
IQ UTILITIES statement
 syntax 338
iq_dummy 302
iq_dummy table 302
IQGOVERN_PRIORITY option 559
IQGOVERN_PRIORITY_TIME option 560
ISOLATION_LEVEL option 561
isysserver system table
 remote servers for Component Integration Services 192

J

jar files
 installing 336
 removing 387
Java
 installing classes 336
 method signatures 175
 removing classes 387
Java table UDF 181
Java VM
 starting 440
 stopping 442
JAVA_LOCATION option 561
JAVA_VM_OPTIONS option 562
jConnect
 disabling support 7
 enabling support 7
join columns
 and data types 302
JOIN_EXPANSION_FACTOR option 562
JOIN_OPTIMIZATION option 563
JOIN_PREFERENCE option 565
JOIN_SIMPLIFICATION_THRESHOLD option 567
joins 419
 deletes 248

- FROM clause syntax 295
 - optimizing 562, 563, 567
 - optimizing join order 579
- K**
- KERBEROS LOGIN statement
 - grant 312
 - revoke 402
- L**
- labels
 - for statements 3, 304
 - LDAP login policy options 29, 160
 - LDAP server configuration object
 - altering 20
 - LDAP server configuration object
 - creating 150
 - dropping 262
 - validating 455
 - LEAVE statement
 - syntax 342
 - LF_BITMAP_CACHE_KB option 567
 - links
 - symbolic 107
 - LOAD TABLE statement
 - HEADER SKIP option 343
 - new syntax 343
 - ON PARTIAL INPUT ROW option 343
 - performance 343
 - QUOTES option 343
 - STRIP keyword 343
 - syntax 343
 - syntax changes 343
 - USING keyword 343
 - LOAD_ZEROLENGTH_ASNULL option 568
 - loads
 - scalability 494
 - LOB variables
 - data type conversion 530
 - LOCK TABLE
 - syntax 362
 - locking
 - tables 362
 - locks
 - releasing with ROLLBACK 413
 - LOG_CONNECT database option 569
 - logical server policies
 - altering 32
 - creating 162
 - defining 153
 - dropping 265
 - logical server policy
 - options 33
 - logical servers
 - altering 22
 - connections 30
 - Login Management
 - POST_LOGIN_PROCEDURE option 601
 - Login Management facility 601
 - login policies
 - altering 24
 - changing 30, 161
 - creating 155
 - dropping 263
 - login policy
 - options 26, 157
 - login processing 601
 - login redirection 32
 - LOGIN_MODE option 571
 - LOGIN_PROCEDURE option 572
 - logins
 - external 131
 - password expiration warning 601
 - See also connections
 - lookup pages
 - maximum 535
 - LOOP statement
 - syntax 364
 - low disk space 123
 - LS policy 33
- M**
- MAIN_RESERVED_DBSPACE_MB option 572
 - materialized view
 - truncating 444
 - materialized views
 - dropping 255
 - MAX_CARTESIAN_RESULT option 573–576
 - MAX_CURSOR_COUNT option 576
 - MAX_HASH_ROWS option 577
 - MAX_IQ_GOVERN_PRIORITY option 559
 - MAX_IQ_THREADS_PER_CONNECTION
 - option 578
 - MAX_IQ_THREADS_PER_TEAM option 578
 - MAX_JOIN_ENUMERATION option 579
 - MAX_PARTITIONED_HASH_MB 580

Index

- MAX_PREFIX_PER_CONTAINS_PHRASE
 - option 581
- MAX_QUERY_PARALLELISM option 581
- MAX_QUERY_TIME option 582
- MAX_STATEMENT_COUNT option 583
- MAX_TEMP_SPACE_PER_CONNECTION
 - option 583
- MDSR encryption algorithm
 - CREATE DATABASE statement 107
- memory
 - prefetching 491
- MESSAGE statement
 - setting DEBUG_MESSAGES option 516
 - SQL syntax 366
- messages
 - creating 163
 - displaying 366
 - dropping 255
- method signatures
 - Java 175
- MIN_PASSWORD_LENGTH option 586
- MIN_ROLE_ADMINIS option 586
- MINIMIZE_STORAGE option 585
- monitor
 - in IQ UTILITIES statement 338
 - setting output file location 587
 - starting and stopping 338
- MONITOR_OUTPUT_DIRECTORY option 587
- monitoring disk space 123
- MPX_AUTOEXCLUDE_TIMEOUT option 588
- MPX_HEARTBEAT_FREQUENCY option 588
- MPX_IDLE_CONNECTION_TIMEOUT option 589
- MPX_LIVENESS_TIMEOUT option 589
- MPX_MAX_CONNECTION_POOL_SIZE option 590
- MPX_MAX_UNUSED_POOL_SIZE option 591
- MPX_WORK_UNIT_TIMEOUT option 591
- multicolumn indexes 142, 147
- multiplex
 - renaming 34
- multiplex databases
 - adding dbspaces 117
 - creating 107
- multiplexes
 - name storage 34
- multirow fetches
 - FETCH statement 283
- multirow inserts 276

N

- named pipes 343
- NEAREST_CENTURY option 592
- newline
 - WD index delimiter 142
- NO RESULT SET clause 165, 175
- NO_SCROLL cursors 240
- NOEXEC option 593
- NON_ANSI_NULL_VARCHAR option 593
- NON_KEYWORDS database option 594
- NOTIFY_MODULUS option 595
- NULL
 - on multicolumn HG index 148
- NULL value
 - in multicolumn HG index 148

O

- ODBC
 - ODBC_DISTINGUISH_CHAR_AND_VAR_CHAR option 595
 - static cursors 240
- ODBC_DISTINGUISH_CHAR_AND_VARCHA
 - R option
 - description 595
- offline
 - dbspaces 9
- ON EXCEPTION RESUME clause
 - stored procedures 598
- ON_CHARSET_CONVERSION_FAILURE
 - option
 - description 596
- ON_ERROR option
 - description 597
- ON_TSQL_ERROR
 - database option 598
- online
 - dbspaces 9
- OPEN statement
 - syntax 369
- optimization
 - defining existing tables and 129
 - MAX_HASH_ROWS option 577
 - MAX_JOIN_ENUMERATION option 579
- option
 - DQP_ENABLED 527
 - DQP_ENABLED_OVER_NETWORK 528
 - ENABLE_LOB_VARIABLES 530
 - MAX_PARTITIONED_HASH_MB 580

- MAX_PREFIX_PER_CONTAINS_PHRASE 581
- MPX_AUTOEXCLUDE_TIMEOUT 588
- MPX_HEARTBEAT_FREQUENCY 588
- MPX_IDLE_CONNECTION_TIMEOUT 589
- MPX_LIVENESS_TIMEOUT 589
- MPX_MAX_CONNECTION_POOL_SIZE 590
- MPX_MAX_UNUSED_POOL_SIZE 591
- NON_ANSI_NULL_VARCHAR 593
- TEXT_DELETE_METHOD 653
- option value
 - truncation 431, 464
- options 302
 - AGGREGATION_PREFERENCE 478
 - ALLOW_SNAPSHOT_VERSIONING 480
 - ASE_FUNCTION_BEHAVIOR 487
 - CIS_ROWSET_SIZE 497
 - compatibility 474
 - CONTINUE_AFTER_RAISERROR 498
 - CONVERSION_ERROR 499
 - CREATE_HG_AND_FORCE_PHYSICAL_DELETE 509
 - CREATE_HG_WITH_EXACT_DISTINCTS 508
 - cursors 465
 - DEBUG_MESSAGES option 516
 - DEDICATED_TASK 517
 - DEFAULT_ISQL_ENCODING 520
 - deprecated 469
 - duration 465
 - ENABLE_ASYNC_IO 530
 - EXTENDED_JOIN_SYNTAX 531
 - finding values 464
 - FLATTEN_SUBQUERIES 636
 - FORCE_DROP 532
 - FP_LOOKUP_SIZE 535
 - FP_LOOKUP_SIZE_PPM 535
 - FP_NBIT_AUTO_LIMIT 536
 - FP_NBIT_ENFORCE_LIMITS 538
 - FP_NBIT_IQ15_COMPATIBILITY 539
 - FP_NBIT_LOOKUP_MB 540
 - FP_NBIT_ROLLOVER_MAX_MB 542
 - general database 469
 - initial settings 468
 - introduction 463
 - list of 477
 - login policies 30, 161
 - MAX_TEMP_SPACE_PER_CONNECTION 583
 - MPX_WORK_UNIT_TIMEOUT 591
 - ODBC_DISTINGUISH_CHAR_AND_VAR_CHAR 595
 - ON_CHARSET_CONVERSION_FAILURE 596
 - ON_ERROR 597
 - ON_TSQL_ERROR 598
 - POST_LOGIN_PROCEDURE 601
 - precedence 465
 - PRESERVE_SOURCE_FORMAT 606
 - RETURN_DATE_TIME_AS_STRING 619
 - REVERT_TO_V15_OPTIMIZER 620
 - ROUND_TO_EVEN 620
 - RV_AUTO_MERGE_EVAL_INTERVAL 622
 - rv_max_active_subfragment_count 623
 - RV_MERGE_NODE_MEMSIZE 623
 - RV_MERGE_TABLE_NUMROWS 625
 - RV_RESERVED_DBSPACE_MBS 625
 - scope 465
 - setting 431, 463
 - setting dbisql options 104
 - setting DBISQL options 104
 - setting temporary 434, 476
 - SNAPSHOT_VERSIONING 624, 627
 - SORT_COLLATION 628
 - sp_iqcheckoptions 464
 - SUBQUERY_CACHING_PREFERENCE 633
 - SUBQUERY_FLATTENING_PERCENT 634
 - SUBQUERY_FLATTENING_PREFERENC E 635
 - SUPPRESS_TDS_DEBUGGING 637
 - SYSOPTIONDEFAULTS system table 464
 - TDS_EMPTY_STRING_IS_NULL 638
 - Transact-SQL 427
 - unexpected behavior 417
 - ORDER BY clause 422
 - OS_FILE_CACHE_BUFFERING option 599
 - OS_FILE_CACHE_BUFFERING_TEMPDB option 600
 - out-of-space conditions
 - preventing 572
 - OUTPUT statement
 - SQL syntax 371
 - owner 3

Index

P

- packages
 - installing 336
 - removing 387
- parallelism
 - backup devices 78
- PARAMETERS statement
 - syntax 374
- partition limit 494
- partitions
 - dropping 52
- password
 - TDS encryption 333
- password encryption
 - TDS 333
- passwords
 - changing 306
 - encryption 333
 - expiration warning 601
 - minimum length 586
- paths
 - relative 107
- performance 302
 - getting more memory 491
- permissions
 - CONNECT privilege 306
- populating 419
- positioned DELETE statement
 - SQL syntax 250
- POST_LOGIN_PROCEDURE option 601
- PRECISION option 602
- predicates 420
- PREFETCH option 603
- PREFETCH_BUFFER_LIMIT option 603
- PREFETCH_BUFFER_PERCENT option 604
- PREFETCH_GARRAY_PERCENT option 604
- PREFETCH_SORT_PERCENT option 605
- prefetching
 - BT_PREFETCH_MAX_MISS 491
- PREPARE statement
 - syntax 375
- prepared statements
 - dropping 271
 - EXECUTE statement 276
- PRESERVE_SOURCE_FORMAT option
 - description 606
- PRINT statement
 - Transact-SQL syntax 378
- privileges, grant
 - ALTER 309
 - DELETE 309
 - INSERT 309
 - LOAD 309
 - REFERENCES 309
 - SELECT 309
 - TRUNCATE 309
 - UPDATE 309
- privileges, revoke
 - ALTER 403
 - DELETE 403
 - INSERT 403
 - LOAD 403
 - REFERENCES 403
 - SELECT 403
 - TRUNCATE 403
 - UPDATE 403
- procedures 375
 - creating 165
 - dropping 255
 - dynamic SQL statements 279
 - executing 278
 - proxy 165
 - RAISERROR statement 381
 - replicating 36
 - result sets 165, 175
 - returning values from 396
 - sa_post_login_procedure 601
 - select from result sets 425
 - sp_droplogin 399
 - sp_iqdroplogin 399
 - Transact-SQL CREATE PROCEDURE statement 172
 - variable result sets 165, 175, 183
- processing by SQL Anywhere 302
- processing queries without 302, 417
- PURGE clause
 - FETCH statement 283
- PUT statement
 - SQL syntax 379
- putting
 - rows into cursors 379

Q

- queries 302
 - for updatable cursors 243
 - improving performance 491
 - LIMIT keyword 417

- processing by SQL Anywhere 417
 - SELECT statement 417
- QUERY_DETAIL option 606
- QUERY_NAME option 607
- QUERY_PLAN option 608
- QUERY_PLAN_AFTER_RUN option 608
- QUERY_PLAN_AS_HTML option 609
- QUERY_PLAN_AS_HTML_DIRECTORY option 611
- QUERY_PLAN_MIN_TIME option 612
- QUERY_PLAN_TEXT_ACCESS option 613
- QUERY_PLAN_TEXT_CACHING option 614
- QUERY_ROWS_RETURNED_LIMIT option 615
- QUERY_TEMP_SPACE_LIMIT option 615
- QUERY_TIMING option 616
- querying tables 302, 417
- QUIT statement
 - syntax 282
- QUOTED_IDENTIFIER option 617
- R**
- RAISERROR statement
 - CONTINUE_AFTER_RAISERROR option 498
 - syntax 381
- raw devices
 - naming 107
- read only
 - locking tables 362
- READ statement
 - syntax 382
- RECOVERY_TIME option 618
- REFERENCES clause 52
- REFRESH TEXT INDEX statement
 - syntax 384
- relative paths 107
- RELEASE SAVEPOINT statement
 - syntax 386
- remote data access 17, 41, 448
 - CIS_ROWSET_SIZE 497
- remote server
 - connecting 332
- REMOVE statement
 - syntax 387
- replication
 - of procedures 36
- RESERVED_KEYWORDS option 618
- RESIGNAL statement
 - syntax 388
- restore operations
 - verifying backups 389
- RESTORE statement
 - COMPATIBLE clause 389
 - improving speed 78
 - syntax 389
 - VERIFY clause 389
 - verifying backups 389
- restoring databases
 - verifying backups 389
- RESTRICT action 205
- result sets
 - SELECT from 425
 - variable 165, 175, 183, 375
- RESUME statement
 - syntax 395
- RETURN statement
 - syntax 396
- RETURN_DATE_TIME_AS_STRING option
 - description 619
- REVERT_TO_V15_OPTIMIZER option 620
- REVOKE CHANGE PASSWORD statement 398
- REVOKE database object privilege statement 403
- REVOKE ROLE statement 404
- REVOKE SET USER statement 407
- REVOKE system privilege statement 408
- Rigndael encryption algorithm
 - CREATE DATABASE statement 107
- role
 - creating 187
 - dropping 266
 - granting 312
 - revoking 404
- roles
 - alter 38
- ROLLBACK statement
 - syntax 413
- ROLLBACK TO SAVEPOINT statement
 - syntax 413
- ROLLBACK TRANSACTION statement
 - syntax 414
 - Transact-SQL 414
- ROLLUP operator 420
- root logical server policy 32
- ROUND_TO_EVEN option
 - description 620
- ROW_COUNT option 621
- row-level versioning 488

Index

- rows
 - deleting from cursors 250
 - inserting using cursors 379
- RV_AUTO_MERGE_EVAL_INTERVAL option 622
- rv_max_active_subfragment_count option 623
- RV_MERGE_NODE_MEMSIZE option 623
- RV_MERGE_TABLE_MEMPERCENT option 624
- RV_MERGE_TABLE_NUMROWS option 625
- RV_RESERVED_DBSPACE_MB option 625
- S**
- sa_conn_properties
 - using 464
- sa_dependent_views system procedure 78
- sa_post_login_procedure 601
- SAVE TRANSACTION statement
 - syntax 416
 - Transact-SQL 416
- SAVEPOINT statement
 - syntax 415
- savepoints
 - name 3
 - RELEASE SAVEPOINT statement 386
 - ROLLBACK TO SAVEPOINT statement 413
 - ROLLBACK TRANSACTION statement 414
 - SAVE TRANSACTION statement 416
- SCALE option 626
- scheduled events
 - WAITFOR statement 458
- scheduling
 - WAITFOR 458
- schema
 - creating 189
- SCROLL cursors 240
- security
 - auditing 488
 - minimum password length 586
- SELECT * 52
- select into 419
- SELECT INTO
 - returning results in a base table 424
 - returning results in a host variable 424
 - returning results in a temporary table 424
- select list 418
 - DESCRIBE statement 251
- SELECT statement 418–421
 - FROM clause syntax 295
 - syntax 417
- separators
 - in WD index 142
- servers
 - altering multiplex 35
 - altering web services 44
 - creating 192
 - creating logical 153
 - deleting logical 264
- services
 - adding 194
- SET CONNECTION statement
 - syntax 429
- SET DESCRIPTOR statement
 - syntax 430
- SET OPTION statement
 - dbisql syntax 476
 - syntax 431, 434
 - using 463
- SET SQLCA statement
 - syntax 435
- SET statement
 - syntax 426
 - Transact-SQL 427
- SET TEMPORARY OPTION statement
 - dbisql syntax 476
 - syntax 431, 434
 - using 463
- set user
 - granting 317
 - revoking 407
- setting dbspaces online 9
- SETUSER statement
 - impersonate 436
- SIGNAL statement
 - syntax 437
- signatures
 - Java methods 175
- SIGNIFICANTDIGITSFORDOUBLEEQUALITY option 628
- SNAPSHOT_VERSIONING option 627
- SORT_COLLATION
 - database option 628
- sp_addmessage 163
- sp_iqcheckoptions system procedure 464
- sp_login_environment procedure 572
- sp_tsqL_environment procedure 572
- specify number of rows 418

- SQL
 - common syntax elements 3
 - statement indicators 5
 - syntax conventions 4
- SQL descriptor area
 - inserting rows using cursors 379
- SQL standards
 - compliance 630, 631
- SQL statements
 - ALTER FUNCTION syntax 16
 - DELETE (positioned) syntax 250
 - MESSAGE syntax 366
 - OUTPUT syntax 371
 - PUT syntax 379
 - UPDATE (positioned) syntax 451
 - WAITFOR syntax 458
- SQL variables
 - creating 232
 - dropping 276
 - SET VARIABLE statement 426
- SQL_FLAGGER_ERROR_LEVEL option 630
- SQL_FLAGGER_WARNING_LEVEL option 631
- SQLCA
 - INCLUDE statement 326
 - SET SQLCA statement 435
- SQLDA
 - allocating memory 5
 - deallocating 237
 - DESCRIBE statement 251
 - Execute statement 276
 - INCLUDE statement 326
 - inserting rows using cursors 379
 - setting 430
 - UPDATE (positioned) statement 451
- standards
 - SQL 1992 compliance 630, 631
 - SQL 1999 compliance 630, 631
 - SQL 2003 compliance 630, 631
- START DATABASE statement
 - syntax 438
- START ENGINE statement
 - syntax 439
- START JAVA statement
 - syntax 440
- starting
 - database servers 439
 - databases 438
 - Java VM 440
- statement indicators 5
- statement labels 3, 304
- statements
 - ALTER FUNCTION syntax 16
 - DELETE (positioned) syntax 250
 - MESSAGE syntax 366
 - OUTPUT syntax 371
 - PUT syntax 379
 - UPDATE (positioned) syntax 451
 - WAITFOR syntax 458
- static cursors
 - declaring 240
- STOP DATABASE statement
 - syntax 441
- STOP ENGINE statement
 - syntax 442
- STOP JAVA statement
 - syntax 442
- stopping
 - Java VM 442
- stopping databases 441
- storage space
 - minimizing 585
- stored procedures
 - creating 165
 - proxy 165
 - sa_dependent_views 78
 - selecting into result sets 425
- STRING_RTRUNCATION option 632
- strings
 - length for database options 431, 464
- STRIP
 - LOAD TABLE keyword 343
- STRIP option 343
- strong encryption
 - CREATE DATABASE statement 107
- subqueries 420
- SUBQUERY_CACHING_PREFERENCE option 633
- SUBQUERY_FLATTENING_PERCENT option 634
- SUBQUERY_FLATTENING_PREFERENCE option 635
- SUBQUERY_PLACEMENT_PREFERENCE database option 636
- SUPPRESS_TDS_DEBUGGING option
 - description 637
- suspended transaction 589
- SWEEPER_THREADS_PERCENT option 638
- symbolic links 107

Index

- syntax
 - common elements 3
- syntax conventions 4
- syntax errors
 - joins 531
- SYSTEM dbspace 302, 417
- system privilege
 - granting 319
 - revoking 408
- system privileges
 - list 320, 410
- system procedures
 - sa_dependent_views 78
- system tables 302
 - PRESERVE_SOURCE_FORMAT 606
 - source column 606
 - SYSFILE 394
- SYSWEBSERVICE system table
 - adding servers 44

- T**
- tab
 - WD index delimiter 142
- table
 - temporary 172
- table constraints 205
- tables 302
 - altering 52
 - altering definition 52
 - creating 205
 - creating proxy 129
 - dropping 255
 - exporting data into files from 371
 - GLOBAL TEMPORARY 205
 - loading 343
 - locking 362
 - temporary 220, 246
 - truncating 444
- TDS
 - password encryption 333
- TDS_EMPTY_STRING_IS_NULL option
 - description 638
- TEMP_DATA_IN_SHARED_TEMP
 - logical server policy option 32
- TEMP_EXTRACT_APPEND option 639
- TEMP_EXTRACT_BINARY option 640
- TEMP_EXTRACT_COLUMN_DELIMITER
 - option 641
- TEMP_EXTRACT_DIRECTORY option 642
- TEMP_EXTRACT_ESCAPE_QUOTES option
 - 643
- TEMP_EXTRACT_NAME1 option 643
- TEMP_EXTRACT_NAME2 option 643
- TEMP_EXTRACT_NAME3 option 643
- TEMP_EXTRACT_NAME4 option 643
- TEMP_EXTRACT_NAME5 option 643
- TEMP_EXTRACT_NAME6 option 643
- TEMP_EXTRACT_NAME7 option 643
- TEMP_EXTRACT_NAME8 option 643
- TEMP_EXTRACT_NAME_n option 643
- TEMP_EXTRACT_NULL_AS_EMPTY option
 - 645
- TEMP_EXTRACT_NULL_AS_ZERO option 646
- TEMP_EXTRACT_QUOTE option 647
- TEMP_EXTRACT_QUOTES option 648
- TEMP_EXTRACT_QUOTES_ALL option 648
- TEMP_EXTRACT_ROW_DELIMITER option
 - 649
- TEMP_EXTRACT_SIZE1 option 650
- TEMP_EXTRACT_SIZE2 option 650
- TEMP_EXTRACT_SIZE3 option 650
- TEMP_EXTRACT_SIZE4 option 650
- TEMP_EXTRACT_SIZE5 option 650
- TEMP_EXTRACT_SIZE6 option 650
- TEMP_EXTRACT_SIZE7 option 650
- TEMP_EXTRACT_SIZE8 option 650
- TEMP_EXTRACT_SIZE_n options 650
- TEMP_EXTRACT_SWAP option 651
- TEMP_RESERVED_DBSPACE_MB
 - database option 652
- TEMP_SPACE_LIMIT_CHECK
 - database option 652
- temporary dbspaces
 - creating 117
- temporary files (Catalog)
 - TEMP_SPACE_LIMIT_CHECK 652
- temporary options 463
- temporary space
 - reserved for IQ store 652
- temporary table 172
- temporary tables 220, 419
 - creating 205
 - declaring 246
- text configuration object
 - altering 68
 - creating 222
 - dropping 272

- TEXT index
 - altering 66
 - creating 223
 - dropping 273
 - text search
 - FROM contains-expression 295
 - TEXT_DELETE_METHOD option 653
 - TIME_FORMAT option 654
 - TIMESTAMP_FORMAT option 655
 - to return one row 418
 - TOP 418
 - TOP_NSORT_CUTOFF_PAGES option 656
 - trailing blanks
 - trimming 343
 - Transact-SQL
 - BEGIN TRANSACTION statement 88
 - COMMIT TRANSACTION 102
 - compatibility options 474
 - CREATE MESSAGE 163
 - CREATE PROCEDURE statement 172
 - CREATE SCHEMA statement 189
 - error handling in 381
 - executing stored procedures 278
 - procedures 172
 - ROLLBACK TRANSACTION statement 414
 - SAVE TRANSACTION statement 416
 - SET statement 427
 - transaction log
 - TRUNCATE statement 444
 - transaction management 102
 - BEGIN TRANSACTION statement 88
 - in Transact-SQL 102
 - transactions
 - committing 102
 - ROLLBACK statement 413
 - ROLLBACK TO SAVEPOINT statement 413
 - ROLLBACK TRANSACTION statement 414
 - SAVE TRANSACTION statement 416
 - SAVEPOINT statement 415
 - suspended 589
 - TRIGGER EVENT
 - syntax 443
 - TRIM_PARTIAL_MBC option 657
 - trimming trailing blanks 343
 - TRUNCATE statement
 - syntax 444
 - TRUNCATE TEXT INDEX statement
 - syntax 445
 - TRUSTED_CERTIFICATES_FILE option 658
 - TSQL_VARIABLES option 658
- ## U
- unexpected behavior 302
 - UNION operation 446
 - unique
 - constraint 205
 - unique indexes 142
 - UPDATE (positioned) statement
 - SQL syntax 451
 - upgrading databases 7
 - USAGE statement
 - grant 323
 - revoke 412
 - user IDs
 - changing passwords 306
 - USER_RESOURCE_RESERVATION option 659
 - user-defined data types
 - altering 13
 - CREATE DOMAIN statement 121
 - dropping 255
 - user-defined functions
 - RETURN statement 396
 - users
 - altering 71
 - creating 230
 - dropping 275, 399
 - USING
 - LOAD TABLE keyword 343
 - USING FILE clause
 - LOAD TABLE statement 343
 - Utilities statement 338
- ## V
- VALIDATE LDAP SERVER statement 455
 - VARCHAR data type
 - converting to compressed format 506
 - variable result sets
 - from procedures 165, 175, 183, 375
 - variables 419
 - creating 232
 - declaring 238
 - dropping 276
 - SET VARIABLE statement 426
 - VERIFY_PASSWORD_FUNCTION option 659
 - verifying backups 389

Index

views

- about 234
- altered tables in 52
- altering 75, 77
- creating 234
- deleting 255
- dependencies 77
- dropping 255
- indexes 146
- invalid 77
- recompiling invalid 77

W

- WAIT_FOR_COMMIT option 663
- WAITFOR statement
 - SQL syntax 458

WASH_AREA_BUFFERS_PERCENT database

- option 662

WD index

- CHAR columns 146
- delimiters 142

WD_DELETE_METHOD option 663

WHENEVER statement

- syntax 459

WHERE clause 419

WHILE statement

- syntax 364
- Transact-SQL 460

wide inserts 276

WITH HOLD clause

- OPEN statement 369

WORD SKIP option 327

- INSERT statement 327