SYBASE®
An SAP® Company

Administration: Database

SAP Sybase IQ 16.0 SP01

DOCUMENT ID: DC01771-01-1601-01
LAST REVISED: May 2013

Contents

Database Administration Workflow 1
Guidelines for Scheduling Data Definition Tasks 1
System Privileges for Data Definition 2
Device Selection 2
Run Database Servers 3
Starting Servers 3
Starting Database Servers 4
The Server Startup Utility 5
Starting Servers from the Windows Start Menu 6
The Server as a Windows Service 6
Automatic Server Startup 6
Command Line Switches 6
Displaying Command Line Options 6
Configuration Files 7
Command Line Example Formatting in SAP Sybase IQ Documentation 8
Required Command Line Options 8
Default Configuration File 8
Configuration File for the Demo Database 8
Naming Restrictions 9
Server and Database Names 9
Command Line Options for Performance 10
Command Line Options That Control Privileges 21
Maximum Catalog Page Size 26
Client/Server Environment Options 27
Forced Recovery Mode Options 30
Starting a Server from Interactive SQL 31
Shared Memory Conflicts 31
Server Activity Logs 32

Naming the Server Log File 34
UNIX Log Files 34
Database Server Shutdown 35
When to Stop and Restart the Server 35
Stopping Servers 35
Permissions Required to Stop the Server 38
Operating System Session Shutdown 38
Starting and Stopping Databases 39
Database Startup Guidelines 39
Stopping Databases 40
Starting the iqdemo Database 40
Connect to Servers and Databases 41
Roadmap for Connections 41
Ways to Connect 42
Connecting to the Demo Database from Interactive SQL 43
How Database Status Affects Local Connections 43
Connecting to a Running Database on a Local Server 43
Connecting to a Database from Interactive SQL on UNIX 44
Connecting from a UNIX System 44
Connecting from a Windows System 45
Connections to Embedded Databases 46
TCP/IP protocol 47
Connecting Using a Data Source 50
Default Connection Parameters 51
Connecting from SAP Sybase IQ Utilities 51
Supported Connection Interfaces 52
Connection Status 53
How Connection Parameters Work 54
Format for Connection Strings 55
How Applications Pass Connection Parameters 55

Connection Parameters in ODBC Data Sources 55
Interactive SQL Connections 56
The Connect Dialog 58
Opening the Connect dialog (Interactive SQL) 58
Drivers for Connections 58
Using Server Classes for Remote Data Access 63
ODBC-Based Server Classes 64
JDBC-Based Server Classes (Deprecated) 77
ODBC Data Sources 79
Where Data Sources Are Held 80
Creating a Data Source from the ODBC Administrator 81
Creating an ODBC Data Source from the Command Line 81
Testing an ODBC Data Source 82
Configuring ODBC Data Sources in ODBC Administrator 82
ODBC Data Sources on UNIX 84
File Data Sources 85
Creating a File Data Source Using the ODBC Administrator 86
File Data Sources and Text Editors 87
The iAnywhere Solutions 16 Oracle ODBC Driver 87
Creating an Oracle DSN on UNIX 89
Creating an Oracle DSN on Windows 89
Creating an Oracle DSN Using IQDSN 90
Database Connections Using OLE DB 90
OLE DB Providers 91
Connections from ADO 92
Connections from Other Databases 93
Avoiding Port Number Conflicts on UNIX 94
How to Test Connections 94
Integrated Logins 95
Using Integrated Logins 96

Security Concerns: Unrestricted Database Access 98
Temporary Public Options Provide Added Security 99
Network Aspects of Integrated Logins 99
Default Integrated Login Users 100
Connection Pooling 100
Temporary Connections 102
Logical Server Configuration 103
Connections in Simplex 104
Connections in Multiplex 104
How to End Connections 105
Connection Logging 106
Create and Manage Databases 107
Create Databases 107
Database Creation with SQL 107
IQ Main Store and IQ Temporary Store Space Management 115
Dropping a Database 116
Disconnecting Other Users From a Database 116
Disconnecting from a Database in Embedded SQL ...117
Disconnecting All Connections from a Database in Interactive SQL 117
Showing System Objects in Interactive SQL 117
Setting Database Options in Interactive SQL 118
Manage Data Storage 119
Space Allocation 119
Types of Dbspaces 119
Catalog Store 121
IQ_SYSTEM_MAIN Dbspace 121
Other User Main Dbspaces 122
IQ Temporary Dbspace 122
IQ Shared Temporary Dbspace 122
IQ Message File Dbspace 124
Space for Databases 124

Space Requirements for IQ Stores and Temporary Stores 124
Sizing Guidelines for Main and Temporary Stores 125
Setting Up Windows Access to Raw Devices 127
Viewing Access Permissions of a RAW Device 128
Setting Permissions to a RAW Device 128
Setting Up Symbolic Links for Raw Devices on Windows 130
Estimating Space and Dbspaces Required 130
Dbspace Management Example 131
Data Storage 137
Dbfile Attributes and Operations 138
Dbspace Attributes and Operations 139
Read-only and Read-write Dbspaces and Files 142
Manage Database Objects 151
SAP Sybase IQ Database Design 151
View Management 151
Creating Views 152
Guidelines for Using Views 153
Guidelines for Modifying Views 154
Permissions on Views 154
How to Delete Views 155
View Information in System Views 155
Table Management 155
Guidelines for Creating Tables 156
Guidelines for Altering Tables 160
Guidelines for Dropping Tables 161
Creating Primary Keys 161
Creating Foreign Keys 162
Table Information in System Views 163
Table Partitions 163
Restrictions 163
Range Partitions 165
Hash Partitions 166

Hash-Range Partitions 167
Index Data 169
Overview of Indexes 169
Index Types Comparison 170
Containment (WD) Index 170
Compare (CMP) Index 172
Date (DATE) and Datetime (DTTM) Indexes 173
Fast Projection (FP) Index 177
High_Group (HG) Index 178
High_Non_Group (HNG) Index 180
Low_Fast (LF) Index 181
TEXT Index 182
Time (TIME) Index 183
Criteria for Choosing Indexes 184
Number of Unique Values in the Index 184
Types of Queries 185
Indexes Recommended by Query Type 186
Indexes Recommended by Column Value 187
Disk Space Usage 188
Data Types in the Index 188
Index Type Combinations 190
Creating Indexes 190
Concurrent Column Indexing 191
Status Messages for Index Loading 191
Adding Column Indexes After Inserting Data 191
Executing Groups of CREATE INDEX Statements 191
Running the Index Advisor 192
Renaming Indexes 192
Viewing Indexes 192
Index Information in System Views 193
Removing Indexes 193
Retaining Indexes When Removing Foreign Key Constraints 193
Optimizing Performance for Joins 194

Enforce Data Integrity 195
Data Integrity Overview 195
How Data Can Become Invalid 195
Rules and Checks for Valid Data 195
Statements That Change Database Contents 196
Data Integrity Tools 196
Statements that Implement Integrity Constraints 197
Column Defaults Encourage Data Integrity 197
Supported Default Values 198
Default Value Restrictions 198
Creating Column Defaults 198
Changing Column Defaults 199
Deleting Column Defaults 199
Supported Column Default Values 199
Date, Time, and Timestamp Defaults 201
USER Defaults 201
The IDENTITY or AUTOINCREMENT Default 202
The NEWID Default 203
The NULL Default 203
String and Number Defaults 203
Constant Expression Defaults 203
Table and Column Constraints 204
UNIQUE Constraints on Columns or Tables 204
IQ UNIQUE Constraints on Columns 204
CHECK Conditions on Columns 205
CHECK Conditions on User-Defined Data Types 206
CHECK Conditions on Columns 206
CHECK Conditions on Tables 207
Removing Check Conditions from Tables 207
Entity and Referential Integrity 207
If a Client Application Breaches Entity Integrity 208
Referential Integrity 208
Concurrent Operations 213

Disabling Referential Integrity Checking 215
Integrity Rules in System Tables 215
Validating Catalog Store Indexes 215
Manage Transactions and Versioning 217
Transactions 217
Viewing Transaction Activity 218
Isolation Levels 221
Transaction Blocking 223
Enabling Connection Blocking 224
Disabling Connection Blocking 224
BLOCKING Option 224
Setting the Blocking Timeout Threshold 225
BLOCKING_TIMEOUT Option 225
Transaction Blocking Deadlocks 226
Versions 234
Table-Level Snapshot Versioning 234
Row-Level Snapshot Versioning 239
Viewing the Versioning Type of a Transaction 245
Temporary Table Versioning 245
Investigating Lock Contention Effects on Performance 246
(deprecated)-contention 246
Checkpoints, Savepoints, and Transaction Rollback 248
Checkpoints 248
Savepoints Within Transactions 249
Transaction Rollback 250
System Recovery 251
How Transaction Information Aids Recovery 251
Concurrency for Backups 252
Cursors in Transactions 252
Cursors and Versioning 252
Cursor Sensitivity 253
Cursor Scrolling 253
Hold Cursors 253
Positioned Operations 254

Message Logging for Cursors 254
Remote Transactions 254
Remote Transaction Restrictions 254
Create Procedures and Batches 255
Procedures 255
Creating Procedures 255
Altering Procedures 256
Calling Procedures 256
Deleting Procedures 256
Privileges to Execute Stored Procedures 256
Returning Procedure Results in Parameters 256
Returning Procedure Results in Result Sets 257
Displaying Procedure Information 257
Displaying Procedure Parameter Information 257
Cursors in Procedures 258
Using IQ UTILITIES to Create Stored Procedures 261
User-Defined Functions 263
Creating User-Defined Functions 263
Calling User-Defined Functions 269
Dropping a user-defined function (SQL) 270
Permissions to Execute User-Defined Functions 271
Granting the ability to execute a user-defined function (SQL) 271
Batches 271
Control statements 275
Compound statements 276
Declarations in compound statements 276
Atomic compound statements 276
Structure of Procedures 277
SQL Statements Allowed in Procedures 277
Parameter declaration for procedures 278
Ways to pass parameters to procedures 279
How to pass parameters to functions 279

Procedure Results 280
Returning a value using the RETURN statement 280
Ways to return results as procedure parameters 281
Information returned in result sets from procedures 283
Returning multiple result sets 284
Variable result sets for procedures 285
Error and warning handling 286
Default handling of errors 286
Error handling with ON EXCEPTION RESUME 288
Default handling of warnings 289
Exception handlers 290
Nested compound statements and exception handlers 292
Transactions and savepoints in procedures 295
Hiding the contents of a procedure, function, trigger, event, or view 295
Statements allowed in procedures, triggers, events, and batches 296
SELECT statements used in batches 297
EXECUTE IMMEDIATE used in procedures, triggers, user-defined functions, and batches 297
Automate Tasks Using Schedules and Events 301
Task automation using schedules and events 301
Events 301
Schedules 301
System events 302
Event handlers 305
Event handlers 308
Retrieving Information About an Event or Schedule 309

Audit Database Events 311
dbtran Database Administration Utility 311
AUDITING Option [database] 315
Troubleshooting Hints 317
Sources of Online Support 317
Solutions for Specific Conditions 317
Decision Flow for Server Recovery and Database Repair 317
Server Operational Issues 318
Database Connection Issues 326
Interactive SQL (dbisql) Issues 328
Resource Issues 328
Processing Issues 335
Performance Issues 339
Troubleshooting Network Communications 339
Using Compatible Protocols 340
Using Current Drivers 340
Powering Down Your Computer Between Restarts 340
Diagnosing the Protocol Stack Layer by Layer . .340
Testing a TCP/IP Protocol Stack 341
Diagnosing Wiring Problems 342
Checking Common Network Communications Problems 342
Diagnostic Tools 343
Restoring to a New Temporary File Topology ….343
The sp_iqstatus Stored Procedure 343
Interpreting Notification Messages 346
The sp_iqcheckdb Stored Procedure 351
Checking Database and Server Startup Option Values 351
Finding the Currently Executing Statement …….351
Logging Server Requests 352
Connection for Collecting Diagnostic Information 356

Diagnosing Communications Issues 356
Reporting Problems to Technical Support 356
Collecting Diagnostic Information Using getiqinfo 357
Information Collected by getiqinfo 358
Correlating Connection Information Between the .srvlog and .iqmsg Files 359
Support Web Site 360
Checklist: Information for Technical Support 361
Appendix: Connection and Communication Parameters Reference 363
Connection Parameters 363
AppInfo Connection Parameter [Appinfo] 364
AutoPreCommit Connection Parameter [AutoPreCommit] 365
AutoStart Connection Parameter [Astart] 366
AutoStop Connection Parameter [Astop] 367
CharSet Connection Parameter [CS] 367
CompressionThreshold (COMPTH) connection parameter 368
CommBufferSize Connection Parameter [CBSize] 369
CommBufferSpace Connection Parameter [CBSpace] 370
CommLinks Connection Parameter [Links] 370
ConnectionName Connection Parameter [CON] 372
ConnectionPool Connection Parameter [CPOOL] 372
DatabaseFile Connection Parameter [DBF] 373
DatabaseName Connection Parameter [DBN] 374
DatabaseSwitches Connection Parameter [DBS] 375

DataSourceName Connection Parameter [DSN] 375
DBKEY Connection Parameter [DBKEY] 376
DisableMultiRowFetch Connection Parameter [DMRF] 376
EngineName Connection Parameter [ENG] 377
EncryptedPassword Connection Parameter [ENP] 377
Encryption Connection Parameter [ENC] 378
Escape Connection Parameter [ESCAPE] 379
FileDataSourceName Connection Parameter [FileDSN] 380
Idle Connection Parameter [IDLE] 380
Integrated Connection Parameter [INT] 381
Language Connection Parameter [LANG] 382
LazyClose Connection Parameter [LCLOSE] ….382
LivenessTimeout Connection Parameter [LTO] 383
LogFile Connection Parameter [LOG] 384
LogicalServer Connection Parameter [LS] 384
LoginRedirection Connection Parameter [REDIRECT] 385
NewPassword (NEWPWD) connection parameter 385
NodeType Connection Parameter 387
Password Connection Parameter [PWD] 387
PrefetchBuffer Connection Parameter [PBUF] ...388
PrefetchRows Connection Parameter [PROWS] 389
RetryConnectionTimeout (RetryConnTO) connection parameter 389
ServerName Connection Parameter [ENG] 390
StartLine Connection Parameter [START] 390
Unconditional Connection Parameter [UNC] 391
Userid Connection Parameter [UID] 391

Network Communications Parameters 392
Broadcast Communication Parameter [BCAST] 393
BroadcastListener Communication Parameter [BLISTENER] 394
ClientPort Communication Parameter [CPort] . . .394
DatabaseName Communication Parameter [DBN] 395
DoBroadcast Communication Parameter [DBROAD] 396
Host Communication Parameter [IP] 397
Identity communication parameter 398
Identity_Password Communication Parameter 398
LDAP Communication Parameter [LDAP] 399
LocalOnly Communication Parameter [LOCAL] 399
LogFile Communication Parameter [LOG] 400
LogFormat Communication Parameter [LF] 400
LogMaxSize (LSIZE) protocol option 401
LogOptions Communication Parameter [LOPT] 402
MaxConnections Communication Parameter [MAXCONN] 403
MaxRequestSize (MAXSIZE) protocol option 403
MyIP Communication Parameter [ME] 404
PreFetchOnOpen Communication Parameter . . .405
ReceiveBufferSize Communication Parameter [RCVBUFSZ] 405
SendBufferSize Communication Parameter [SNDBUFSZ] 406
ServerPort Communication Parameter [PORT] 406
Sessions Communication Parameter 407
TDS Communication Parameter 408

Timeout Communication Parameter [TO] 409
VerifyServerName Communication Parameter [Verify] 409
Appendix: SQL Statements and Options Reference 411
Database Options 411
Scope and Duration of Database Options 411
DEDICATED_TASK Option 412
LOG_CONNECT Option 412
MIN_PASSWORD_LENGTH Option 413
VERIFY_PASSWORD_FUNCTION Option 413
SQL Statements 416
ALTER DBSPACE Statement 416
ALTER INDEX Statement 420
ALTER LOGICAL SERVER Statement 422
ALTER LS POLICY Statement 424
ALTER TABLE Statement 426
CHECKPOINT Statement 440
COMMIT Statement 441
CREATE DATABASE Statement 443
CREATE DOMAIN Statement 453
CREATE INDEX Statement 455
CREATE LOGICAL SERVER Statement 463
CREATE LS POLICY Statement 465
CREATE TABLE Statement 467
DROP Statement 484
DROP LOGICAL SERVER Statement 487
GRANT INTEGRATED LOGIN Statement 488
IQ UTILITIES Statement 488
LOCK TABLE Statement 492
ROLLBACK Statement 494
SAVEPOINT Statement 495
SET OPTION Statement 496
STOP DATABASE Statement [Interactive SQL] 498
Index 501

Administration Workflow

Creating an SAP® Sybase® IQ database is part of a larger setup process that begins with installation and ends when your database is available to users.

1. Install and configure SAP Sybase IQ.

   Install both the client and server environments. See the *Installation and Configuration Guide* guide.
2. Start and connect to a database server.

   Enable database control and open a communication channel to the database. Use the **start_iq** utility, Sybase Control Center, the Windows Start menu, or a configuration file.
3. Create an IQ database.

   Create both the IQ store and the catalog store. You may use Sybase Control Center, the SQL statement **CREATE DATABASE**, or the **iqinit** utility.
4. Create the tables in your IQ database.

   Use the **CREATE TABLE** statement or the Sybase Control Center Create Table wizard.
5. Create indexes for the tables.

   Use the **CREATE INDEX** statement or the Sybase Control Center Create Index wizard. You can also create certain indexes automatically when you create your tables.
6. Load data into the tables.

   Use the **LOAD TABLE** statement to bulk-load data from files, or use the **INSERT** statement to extract rows of data from an existing database. See *Administration: Load Managementt*.

Guidelines for Scheduling Data Definition Tasks

Once the database exists and other users have access to it, you must perform additional data definition operations, such as adding or modifying tables or indexes.

Schedule data definition operations for times when database usage is low. All other users are blocked, though only briefly, from reading or writing to a table while you are creating or altering that table.

For more information on concurrency rules during data definition, see *Table Locks for DDL Operations*.

System Privileges for Data Definition

You must have the appropriate system privileges to perform data definition tasks.

- With the SERVER OPERATOR system privilege, you can perform all data definition tasks. You also can grant this system privilege to other users to perform specific tasks. This includes the ability to grant the SERVER OPERATOR system privilege to other users.
- To create any database object, you need the CREATE ANY OBJECT system privilege for that type of object.
- When you create an object, you become its owner. The owner of an object automatically has the system privilege to perform all operations on that object, and to grant other users the system privilege to update the information in a table.

Users with administrative rights to the required system privileges and object owners can grant required system privileges to individual users. You can also use the **-gu** command line option to set the permission level required to create or delete a database.

Device Selection

Store databases and database objects on devices. On all platforms, these devices can be operating system files, or they can be portions of a disk, called raw partitions. When you create a database, SAP Sybase IQ determines automatically whether it is a raw partition or a disk file.

In a production environment, raw partition installations may improve processing performance and recovery capabilities. File systems, however, offer easier device management, and may be preferable in a development environment.

Note: The catalog store and the transaction log cannot be on a raw partition.

Run SAP Sybase IQ Database Servers

Each SAP Sybase IQ database runs on a server. Depending on your platform and administration tool, SAP Sybase IQ offers multiple ways to start and stop a server.

Starting Servers

Different ways to start a server.

Server Startup Methods for Any Platform
Several methods start the database server.

To Start the Server From ...	Do This ...
Server startup utility	Enter a **start_iq** command on the command line. Syntax: `start_iq [ options ]` If you enter several command line options, you must enter them all on one line (without carriage returns). For ease of use, specify the database name when you start the server. This starts the database and server together. Omit the database name to start the server without starting a database. If you omit the database name, you must name the server explicitly using the **-n** server switch. Use this method when you create or restore a database, or when you only want to control starting and stopping the server, leaving database use to client software.
Configuration file	Start the server and the `iqdemo` database with a SAP Sybase-provided configuration file.
Server startup command	Place a server startup command in a shortcut or desktop icon.
ODBC datasource	Include a server start line in the datasource.
Utility command	Include a server start line in a utility command.
Interactive SQL	Issue a SQL command from Interactive SQL to start an additional server.

To Start the Server From ...	Do This ...
Sybase Control Center	To start in Sybase Control Center, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

Windows Server Startup Methods

Several methods start a Windows server.

To start the server from...	Do this...
Command line	Run the **stop_iq** or **dbstop** command.
Desktop	Right-click the **server icon** in the system tray, and choose **Start server name**.
Windows Start menu	Click **Start > All Programs > Sybase > Sybase IQ 16.0 > Start Sybase IQ Demo Database**.
Service Manager	If you started the server as a Windows service: 1. On the **Control Panel**, choose **Administrative Tools > Services**. 2. Choose the SAP Sybase IQ service, then click **Start the service**.

Starting Database Servers

The first step in running SAP Sybase IQ is to start the database server.
Start the server in any of these ways:

- Using the server startup utility, **start_iq**.
- Using Sybase Control Center. To start in Sybase Control Center, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.
- From the Windows Start menu.
- Start the server and the `iqdemo` database with a Sybase-provided configuration file.
- Place a server startup command in a shortcut or desktop icon.

  Note: You can also configure Windows systems to start an IQ server automatically (as a Windows Service) when the system is booted.
- Include a server start line in an ODBC datasource.
- Include a server start line in a utility command.

- Issue a SQL command from Interactive SQL to start an additional server.

The Server Startup Utility

The startup utility, **start_iq**, runs on all platforms, and ensures that required parameters are set correctly, with a few documented exceptions.

In examples that include several command line options, they are, for clarity, shown on separate lines, as they could be written in a configuration file. If you enter them directly on a command line, you must enter them all on one line (that is, without any carriage returns).

You can choose from many command line switches to specify such features as permissions required to start a database or stop the server, and the network protocols to use. Command line options are one way to tune SAP Sybase IQ behavior and performance.

Startup of Database with Database Server

For ease of use, start the database and server together, by specifying the database name when you start the server.

By default, the server takes its name from the database name, or you can specify a different name for the server.

Startup of Database Server Without Database Startup

To start the server without starting any database, omit the database file from the **start_iq** command and specify a server name.

If you omit the database name, you must name the server explicitly using the **-n** server switch. Use this method when you create or restore a database, or when you only want to control starting and stopping the server, leaving database use to client software.

Starting the Server Using the Startup Utility

The **start_iq** command starts the named server as a background process, starts the named database if you specify it, and sets all required startup options.

1. Change to a writable directory.
2. At the system prompt, enter:

```
start_iq servername [ database ]
```

If you do not specify the database, you must use **-n** <*server name*> or the server does not start. In the preceding example, the server starts on the default port, 2638.

Server Log

Once the server starts, it sends a message to the window or console where you started the server indicating that the server is running.

The server also displays other information about your server environment, as well as `possible problems` messages if it fails to start.

All server messages are written to the server log. By default, the environment variable %IQLOGDIR16% is set by the installation on Windows platforms, and the server log is in the path %IQLOGDIR16%*servername.nnnn*.srvlog, where *nnnn* is the number of times the server has been started. You can also use the **-o** startup option to name the server log.

Starting Servers from the Windows Start Menu

Certain methods for starting the database server are specific to Windows systems. You can also use a generic method to start the database server.
The easiest way to start the server on Windows is from the Start menu. Select **Programs > Sybase > Sybase IQ 16.0**.

From here, you can start the SAP Sybase IQ Demo database and **Interactive SQL**.

You can also place databases of your own in the Program group.

The Server as a Windows Service

You can run the server as a service under Windows, which allows it to keep running even when you log off the machine.

For details of this and other Windows-specific features, see the *Installation and Configuration Guide* guide.

Automatic Server Startup

Use SAP Sybase IQ Service Manager to define a service that starts an SAP Sybase IQ server.

You can then configure the service to automatically start the server whenever the host is started. The service may start either non-multiplex or multiplex servers.

Command Line Switches

Command line switches define your SAP Sybase IQ environment.

For a complete list of command line switches and full reference information, see *Utility Guide > start_iq Database Server Startup Utility*.

Some of the values you can set with command line options can also be changed with the **SET OPTION** command.

Displaying Command Line Options

Command line switches are case sensitive.
To display all of the available command line switches, enter this command at the operating system prompt:

```
start_iq -h
```

Configuration Files

You can store sets of command line switches in configuration files.

If you use an extensive set of command line switches, you can store them in a configuration file, and invoke that file in a server command line. Specify switches in the configuration file in the same way as on the command line, except in the file, you can enter switches on multiple lines.

Configuration File List

SAP Sybase IQ provides configuration files that you can use as templates.

Table 1. Configuration Files

File Name	Location	Use
`default.cfg`	`$IQDIR16/scripts` (UNIX), `%IQDIR16%\scripts` (Windows)	Generic configuration file. This file is used for default options for **start_iq** and multiplex startup. SAP Sybase IQ copies `default.cfg` into each new database directory and renames it `params.cfg`. Any changes that you make to `default.cfg` (in the `scripts` directory) are inherited by all databases that are created after the file is changed.
`iqdemo.cfg`	`$IQDIR16/data/demo` (UNIX), `%IQDIR16%\data\demo` (Windows)	Starts the demo database, setting startup switches to the recommended defaults.

Examples of Configuration Files

You can use the predefined configuration files as templates to create your own.

For example, the following configuration file starts the database **mydb.db** on the database server named `Elora`, with a 32MB cache and a 20-minute checkpoint interval. It allows anyone to start or stop databases and load data, limits user connections to 10, has a catalog page size of 4096 bytes, has a default client connection timeout of 72 hours, uses TCP/IP as a network protocol, and has a specified port number of 1870:

```
-n Elora -c 32m -gc 20
-gd all -gl all -gm 10 -gp 4096 -ti 4400 -x tcpip(port=1870) path
\mydb.db
```

You can execute all these command line options using:

```
start_iq @mydb.cfg
```

Note: When you stop the server with the **DBSTOP** command, you must specify the same parameters as when you started the server; if you use a configuration file to start the server ensures that you can find these parameters when you need them to stop the server.

Command Line Example Formatting in SAP Sybase IQ Documentation

For clarity, examples throughout SAP Sybase IQ documentation show multiple command line switches on separate lines as you can write them in a configuration file.

If you enter them directly on a command line, you must enter them all on one line.

Required Command Line Options

Not all command line switches are optional.

While most of the command line switches described in *Utility Guide > start_iq Database Server Startup Utility* are optional, you must specify the **-n** switch to run SAP Sybase IQ effectively.

Note: On all 32-bit platforms, **-c 32M** is recommended. On all 64-bit platforms, **-c 48M** is recommended.

If you use TCP/IP to connect to the server, include network connection parameters as well. If you start the server without the parameter **-x 'tcpip(port=nnnn)'**, the server uses the default TCP/IP port number, 2638. If you specify a port number that is already in use, the server fails to start.

Default Configuration File

The default configuration file (`default.cfg`) contains all of the required startup switches.

The `default.cfg` file:

- Provides the source for the `params.cfg` file used by the UNIX **start_iq** command
- Starts servers for Windows services

To override any switch except for *-n servername* in configuration files, specify new switches on the **start_iq** command line.

Configuration File for the Demo Database

The `iqdemo.cfg` file, which you use to start the demo database, sets startup switches to the recommended defaults.

You can automatically create this file when you create the demo database using scripts provided by the SAP Sybase IQ.

Naming Restrictions

Do not use hyphenated names or reserved words for database names, user identifiers, or server names, even enclosed in quotation marks.

For example, the following are not allowed:

```
grant

june-1999-prospects

"foreign"
```

For a complete list of reserved words (keywords), see *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements > Keywords > Reserved Words*.

Server and Database Names

Use the **-n** command line switch as a server switch (to name the server). This prevents you from unintentionally connecting to the wrong server.

The server and database names are among the connection parameters that client applications can use when connecting to a database. On Windows, the server name appears on the desktop icon and on the title bar of the server window.

Note: While you can start more than one database, SAP Sybase strongly recommends that you run only one database per IQ server. If you must run multiple databases, start each one on a separate IQ database server, and on a different port.

Default Server Names

If no server name is provided, the first database started provides the default server name.

Case-Sensitivity and Naming Conventions

Server names and database names are case-insensitive on Windows, and case-sensitive on UNIX.

Adopt a set of naming conventions for your servers and databases, as well as for all other database objects, that includes a case specification. Enforcing naming conventions can prevent problems for users.

Database Names

You can name databases by supplying a **-n** switch following the database file.

For example, this command line starts a database and names it:

```
start_iq -n MyServer mydb.db -n MyDB
```

Naming a database lets you use a nickname.

Server Names

Name a server by supplying a **-n** switch before the first database file. The rest of the parameters are added from the `default.cfg` file.

For example, this command starts a server named `Cambridge_iqdemo` and the `iqdemo` database on that server:

```
start_iq -n Cambridge_iqdemo iqdemo.db
```

Each server name must be unique across the local area network (domain). This prevents you from unintentionally connecting to the wrong server. The host name and port number combination does not uniquely identify the server. Appending a unique identifier to the server name is a useful convention. Assigning unique server names is especially important in a multiuser, networked environment where shared memory is used for local database connections. This convention ensures that all users can connect to the correct database, even when other databases with the same name have been started on other host systems.

To allow SAP Sybase IQ to locate the server regardless of the character set in use, include only seven-bit ASCII (lower page) characters in the server name.

Specifying a server name lets you start a database server with no database loaded. This command starts a server named `Galt` with no database loaded:

```
start_iq -n Galt -gm 10 -gp 4096
```

Note: Although you can start a server by relying on the default server name, it is a better practice to include both the server name and the database name, and to make the two names different. This approach helps users distinguish between the server and the databases running on it. You must specify the server name to start the server without starting a specific database.

Command Line Options for Performance

Several command line options can affect database server performance.

Most of the performance command line options control resources for operations on the IQ store, which can have a major impact on performance. Options that affect only the resources available for operations on the catalog store may have a minor impact on overall performance.

See also

- *Database Options* on page 411

Memory Options

SAP Sybase IQ uses memory for a variety of purposes.

- Buffers for data read from disk to resolve queries
- Buffers for data read from disk when loading from flat files
- Overhead for managing connections, transactions, buffers, and database objects

IQ Buffer Cache Size Controls

The IQ Buffer Cache Size options, as well as other options you can set once the server is running, determine how much memory is available.

The default IQ buffer cache sizes of 16MB for the main cache and 8MB for the temporary cache are too low for any active database use.

Set the buffer cache sizes for the IQ main and temporary stores in one of two ways:

- (Recommended) To set buffer cache sizes server-wide for the current server session, specify the database startup utility **start_iq** options **-iqmc** (main cache size) and **-iqtc** (temp cache size).
- To set cache sizes for a database, use the **sa_server_option** stored procedure with **main_cache_memory_mb** or **temp_cache_memory_mb** parameters.

If you set IQ buffer cache sizes higher than your system can accommodate, however, SAP Sybase IQ cannot open the database.

The server options (**-iqmc** and **-iqtc**) also let you use as much memory as your system allows, the only limit being the amount of physical memory on the machine. For this reason, on 64-bit systems, use **-iqmc** and **-iqtc**, which do not override the settings made by **sa_server_option**.

The cache sizes set by **-iqmc** and **-iqtc** apply to all databases started until the server is shut down. So, for example, if you set both **-iqmc** and **-iqtc** to 500 (MB) and start one database at server startup and another database later on the same server, you need at least 2GB available for the two main and two temp caches.

-iqmc iqsrv16 Server Option

Specifies the main IQ store cache size, in MB.

Syntax

-iqmc *size*

Default

64MB

Remarks

The switch overrides the default value of 64MB, and applies to all databases started from the time the server is started until the server is shut down. In other words, if you start one database at server startup and another later, you need 2 * **-iqmc** available for the main cache.

Large memory requirements represent one third of all available physical memory. To ensure adequate memory for the main store, set the **–iqmc** startup parameter to one third of all available physical memory.

Always specify the size value, without including the units of measurement; for example, specify **-iqmc** `32` rather than **-iqmc** `32MB`. If you specify the unit of measurement, **start_iq** ignores this switch, unlike SQL Anywhere, which requires a unit of measurement.

Note: Do not run multiple databases with an SAP Sybase IQ server.

-iqtc iqsrv16 Server Option
Specifies IQ temporary store cache size, in MB.

Syntax
-iqtc *size*

Default
64MB

Remarks
The switch overrides the default value of 64MB, and applies to all databases started from the time the SAP Sybase IQ server is started until the server is shut down. In other words, if you start one database at server startup and another later, you need 2 * **-iqtc** available for the temp cache. In general, avoid running multiple databases with an SAP Sybase IQ server.

Large memory requirements represent one third of all available physical memory. To ensure adequate memory for the IQ temporary store cache, set the **–iqtc** startup parameter to one third of all available physical memory.

Always specify the size value, without including the units of measurement; for example, specify **-iqtc** `32` instead of **-iqtc** `32MB`. If you specify the unit of measurement, **start_iq** ignores this switch, unlike SQL Anywhere, which requires a unit of measurement.

Ensure that the IQ_SYSTEM_TEMP dbspace is at least as large as **-iqtc**.

IQ Buffer Partitioning Control
By default, buffer partitioning based on the number of CPUs is enabled. Adjust the partitions to improve load or query performance.

-iqpartition iqsrv16 Server Option
Sets the number of IQ main and temp buffer cache partitions.

Syntax
-iqpartition *num*

Remarks
Specifies the number of partitions in the IQ main and temp buffer caches. Must be a power of 2. By default, allowed values are: 0 (default), 1, 2, 4, 8, 16, 32, 64, 128, 256. By default, SAP Sybase IQ computes the number of partitions automatically as *number_of_cpus/8*, rounded to the nearest power of 2, up to a maximum of 64. You may be able to improve performance by adjusting the number of cache partitions. The **-iqpartition** switch sets this value for an SAP Sybase IQ server, and overrides the value set by the `Cache_Partitions` database option.

- Excludes jConnect™ for JDBC™ system objects from the database. To use the jConnect JDBC driver to access system catalog information, you need jConnect catalog support (installed by default). When you specify this option, you can still use JDBC, as long as you do not access system information. You can add jConnect support at a later time using the **ALTER DATABASE** statement.

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 `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 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
```

Concurrent User Switches
Your license sets the absolute number of concurrent users. The required **-gm** switch limits the number of concurrent user connections on a particular server.

The **-gn** switch sets the number of execution threads that are used for the catalog store and connectivity while running with multiple users. It applies to all operating systems and servers.

On Windows, **start_iq** calculates the value of this parameter and sets it using this formula:

```
gn_value >= gm_value * 1.5
```

Set the **-gn** value to at least 1.5 times the value of **-gm**. Specify a minimum of 25. The total number of threads cannot exceed a platform-specific maximum; see **-iqmt** *num* for details.

-gm iqsrv16 database server option
Limits the number of concurrent connections to the database server.

Syntax
```
iqsrv16 -gm integer ...
```

Default
The default value for the personal server is 10. The default value for the network database server is 32766, though this number will be reduced by internal temporary connections utilized by the server during operation.

Applies to
All operating systems and database servers.

Remarks
Defines the connection limit for the server. If this number is greater than the number that is allowed under licensing and memory constraints, it has no effect. Computer resources typically limit the number of connections to a network server to a lower value than the default.

The database server allows one extra DBA connection above the connection limit to allow a user with the DROP CONNECTION system privilege to connect to the database server and drop other connections.

-gn iqsrv16 Server Option

Sets the number of execution threads that are used for the catalog store and connectivity while running with multiple users.

Syntax

-gn *integer*

Remarks

This parameter applies to all operating systems and servers. Each connection uses a thread for each request, and when the request is completed, the thread is returned to the pool for use by other connections. As no connection can have more than one request in progress at one time, no connection uses more than one thread at a time.

An exception to this rule is if a Java application uses threads. Each thread in the Java application is a database server execution thread.

On Windows, specify this parameter in **start_iq**. To calculate its value, use:

```
gn_value >= gm_value * 1.5
```

SAP Sybase recommends that you set the **-gn** value to at least 1.5 times the value of **-gm**. Specify a minimum of 25. The total number of threads cannot exceed a platform-specific maximum; see **-iqmt** *num* for details.

Concurrent Query Switch

To limit the number of user connections to fewer than your license allows, use the **-iqgovern** switch to control query use.

The **-iqgovern** switch controls the number of concurrent queries on a particular server.

Use the **-iqgovern** switch to help IQ optimize paging of buffer data out to disk and avoid overcommitting memory. The default value of **-iqgovern** is (2 x the number of CPUs) + 10. You may need to experiment to find an ideal value. For sites with large numbers of active connections, set **-iqgovern** slightly lower.

-iqgovern iqsrv16 Server Option

Sets the number of concurrent queries allowed by the server.

Syntax

-iqgovern *num*

Remarks
The number of concurrent queries is not the same as the number of connections. A single connection can have multiple open cursors. **-iqgovern** can help SAP Sybase IQ optimize paging of buffer data out to disk and avoid overcommitting memory. The default value of this switch is equal to 2 times the number of CPUs on your machine, plus 10. You may find that another value, such as 2 times the number of CPUs plus 4, provides better throughput, especially when large numbers of users are connected.

Wired Memory Switch
The **-iqwmem** switch creates a pool of "wired" memory on certain UNIX platforms only.

Warning! Use this switch only if you have enough memory to dedicate some of it for this purpose. Otherwise, you might cause serious performance degradation.

-iqwmem iqsrv16 Server Option
Creates a pool of "wired" memory on HP and Sun UNIX systems.

Syntax
-iqwmem *size*

Remarks
This memory is locked down so it cannot be paged by the operating system. Specify the memory size, in MB. Use this switch only if you have enough memory to dedicate for this purpose. Otherwise, you may cause serious performance degradation.

Number of Processing Threads
SAP Sybase IQ assigns varying numbers of kernel threads to each user connection, based on the type of processing being done by that process, the total number of threads available, and the setting of various options. Increasing the number of threads might improve performance.

Use the **-iqmt** switch to set the number of processing threads that SAP Sybase IQ can use.

-iqmt iqsrv16 Server Option
Specifies the number of SAP Sybase IQ threads to create.

Syntax
-iqmt *num*

Remarks

The default is 60 threads per CPU for the first 4 CPUs and 50 threads per CPU for the remainder, with 3 more for system use, plus threads needed for database connections and background tasks. For example, on a system with 12 CPUs and 10 connections: `60*4 + 50*(numCPUs - 4) + numConnections + 6 = 656`.

The minimum value of num is `num_conn + 3`.

The total number of server threads cannot exceed 4096 on 64-bit platforms, or 2048 on 32-bit platforms.

Number of Processors

On a multiprocessor machine, use the **-gt** option, to set the number of processors used by the database server for catalog store operations.

By default, all available processors are used.

-gt iqsrv16 database server option

Sets the maximum number of physical processors that can be used (up to the licensed maximum). This option is only useful on multiprocessor systems.

Syntax

```
iqsrv16 -gt num-processors ...
```

Allowed values

- **num-processors –** This integer can be a value between 1 and the minimum of:

  the number of physical processors on the computer
  the maximum number of CPUs that the server is licensed for if CPU-licensing is in effect

If the -gt value specified lies outside this range, the lower or upper limit is imposed.

Remarks

With per-seat licensing, the network database server uses all CPUs available on the computer. With CPU-based licensing, the network database server uses only the number of processors you are licensed for. The number of CPUs that the network database server can use may also be restricted by your SAP Sybase IQ edition.

When you specify a value for the -gt option, the database server adjusts its affinity mask (if supported on that hardware platform) to restrict the database server to run on only that number of physical processors. If the database server is licensed for *n* processors, the server, by default, runs on all logical processors (hyperthreads and cores) of *n* physical processors. This behavior can be further restricted with the -gtc option.

The -gt option cannot be used with the -gta option.

Catalog Store Cache Size

Use the **-c** switch to set the amount of memory in the cache for the catalog store.

The **start_iq** command, and the `iqdemo.cfg` and `default.cfg` configuration files set the **-c** parameter to 48MB on 64-bit systems and 32MB on 32-bit systems.

Any cache size smaller than 10000 is assumed to be in KB (1K =1024 bytes); any cache size 10000 or greater is assumed to be in bytes. You can also specify the cache size as *n*K or *n*M.

If you start the server without using **start_iq**, `iqdemo.cfg`, or `default.cfg`, the default initial cache size is computed based on the amount of physical memory, the operating system, and the size of the database files. The database server takes additional cache for the catalog when the available cache is exhausted.

Warning! To control catalog store cache size, you must do either of the following, but not both, in your configuration file (`.cfg`) or on the UNIX command line for server startup:

- Set the **-c** parameter, or,
- Set specific upper and lower limits for the catalog store cache size using the **-cl** and **-ch** parameters

Do not specify other combinations of these parameters.

The cache size for the IQ store does not rely on the catalog cache size.

-c iqsrv16 Server Option

Sets initial memory reserved for caching catalog store pages and other server information.

Syntax

-c *cache-size[k| m| g| p]*

Remarks

The database server uses extra memory for caching database pages if the memory is set aside in the cache. Any cache size less than 10000 is assumed to be KB (1K = 1024 bytes). Any cache size 10000 or greater is assumed to be in bytes. You can also specify the cache size `nK`, `nM` or `nP` (1M = 1024 KB), where **P** is a percentage of the physical system memory.

In the `default.cfg` file, the default value of **-c** and **start_iq** is 32MB (**-c 32M**) for Windows platforms, and 48MB (**-c 48M**) for UNIX platforms.Use this default or set **-c** to a higher value.

You can use % as an alternative to P, but as most non-UNIX operating systems use % as an environment variable escape character, you must escape the % character. For example, to use 20 percent of the physical system memory, specify:

```
start_iq -c 20%% ...
```

Do not use **-c** in the same configuration file or command line with **-ch** or **-cl**. For related information, see the **-ch cache-size** option and the **-ca 0** option.

Initial Catalog Store Cache Size Without Specifying start_iq Server Option -c

If you omit a value for the **start_iq** server option **-c** switch (either on the command line or using the **start_iq** default), the database server computes the initial catalog store cache allocation

The server computes the cache allocation as follows:

- The database server uses 32MB as the minimum default cache size.
- The database server computes a runtime-specific minimum default cache size, which is the lesser of:
  - 25% of the physical memory of the machine, or,
  - The sum of the sizes of the main database files specified on the command line. Additional dbspaces apart from the main database files are not included in the calculation. If no files are specified, this value is zero.
- The database server allocates the greater of the two values computed.

Number of CPUs Switch

The **-iqnumbercpus** switch on the SAP Sybase IQ startup command specifies the number of CPUs available to IQ, overriding the physical number of CPUs for resource planning purposes.

The value of the parameter defaults to the total number of CPUs, but the range of available values is 1 through 512.

Note: Use **-iqnumbercpus** only on machines:

- With Intel® CPUs and hyperthreading enabled. Set **-iqnumbercpus** to the number of CPUs available.
- Where an operating system utility has been used to restrict SAP Sybase IQ to a subset of the CPUs within the machine

Setting **-iqnumbercpus** higher than the number of available CPUs might affect performance.

Options That Affect Timing

Several command line options control when certain database server events occur.

Checkpoint Interval

SAP Sybase IQ uses checkpoints to generate reference points and other information necessary for recovering databases.

Use the **-gc** switch to set the maximum number of minutes the database server runs without performing a checkpoint.

When a database server is running with multiple databases, the checkpoint time specified by the first database started is used unless overridden by the **-gc** switch. If a value of 0 is entered, the default value of 20 minutes is used.

Recovery Time

The **-gr** parameter lets you set the maximum number of minutes that the database server takes to recover from system failure.

When a database server runs with multiple databases, the recovery time specified by the first database started is used unless **-gr** overrides that value.

-gc iqsrv16 database server option
Sets the maximum interval between checkpoints.

Syntax
```
iqsrv16 -gc minutes ...
```

Default
60 minutes

Allowed values

- **minutes –** The default value is the setting of the checkpoint_time database option, which defaults to 60 minutes. If a value of 0 is entered, the default value of 60 minutes is used.

Applies to
All operating systems and database servers.

Remarks
Use this option to set the maximum length of time, in minutes, that the database server runs without doing a checkpoint on each database.

Checkpoints generally occur more frequently than the specified time.

Network Performance Options
Use the **-gb** (database process priority on Windows), and **-p** (maximum packet size) options to help you tune network performance.

-gb iqsrv16 database server option
Sets the server process priority class.

Windows syntax
```
iqsrv16 -gb { idle | normal | high | maximum } ...
```

Unix syntax
```
iqsrv16 -gb level ...
```

Allowed values

- **Unix –** On Unix, the *level* is an integer from -20 to 19. The default value on Unix is the same as the nice value of the parent process. Lower *level* values represent a more favorable scheduling priority. All restrictions placed on setting a nice value apply to the -gb option. For example, on most Unix platforms, only the root user can lower the priority level of a process (for example, changing it from 0 to -1).

- **Windows –** On Windows, normal and high are the commonly-used settings. The value idle is provided for completeness. The value maximum may interfere with the running of your computer.

Applies to
Windows, Unix.

-p iqsrv16 database server option
Sets the maximum size of communication packets.

Syntax
```
iqsrv16 -p integer ...
```

Default
7300 bytes

Applies to
All operating systems and database servers.

Remarks
The minimum value is 500 bytes and the maximum value is 65535.

You can change the communication buffer size for a connection by setting the CommBufferSize (CBSIZE) connection parameter.

Command Line Options That Control Privileges

The database server start-up command **start_iq** includes options that set the privilege level of some database and server functions.

Switches That Start and Stop Databases
The **-gd** option lets you limit the users who can start or stop a database on a running server to those with a certain level of privilege in the database to which he or she is already connected:

- **DBA** – (default value) only users with SERVER OPERATOR system privilege can start an extra database.
- **ALL** – (default in **start_iq** and `default.cfg`) any user can start and stop databases. This setting means that the DBA does not need to issue **START DATABASE** commands. Users still need the privileges to access a particular database once he or she has started it.
- **NONE** – no one can start or stop a database from Interactive SQL on a running server.

Note: If **-gd ALL** is not set when you start the server, only a user with the SERVER OPERATOR system privilege can start additional databases on that server. This means that users cannot connect to databases that are not already started, either at the same time as the server, or since then by a user with the SERVER OPERATOR system privilege. However, it

also lets a user without the SERVER OPERATOR system privilege stop a database. For this reason, you may want to change this setting to DBA on production databases.

Switches That Create and Delete Databases

The **-gu** option limits the users who can create and drop databases to those with a certain level of privilege in the database to which he or she is connected.

- **DBA** – only users with SERVER OPERATOR system privilege can create and drop databases.
- **ALL**(default) – any user can create and drop databases.
- **NONE** – no user can create or drop a database.
- **UTILITY_DB** – only those users who can connect to the `utility_db` database can create and drop databases.

Stop Server Switch

The **-gk** option limits the users who can shut down a server with the **dbstop** utility or **STOP ENGINE** command:

- **DBA** (default) – only users with SERVER OPERATOR system privilege can stop the server.
- **ALL** – any user can stop the server.
- **NONE** – no user can shut down the server with the **dbstop** utility or **STOP ENGINE** command.

Switches That Load and Unload Databases

The **-gl** option limits the users who can load data using **LOAD TABLE** to users with a certain level of privilege in the database.

- **DBA** – any user with the LOAD ANY TABLE, ALTER ANY TABLE or ALTER ANY OBJECT system privilege can load data.
- **ALL** (default for **start_iq** and `default.cfg`) – any user can load data.
- **NONE** – data cannot be loaded.

-gd iqsrv16 database server option

Sets the privileges required to start or stop a database on a running database server.

Syntax

```
iqsrv16 -gd { DBA | all | none } ...
```

Allowed values

- **DBA –** Only users with the SERVER OPERATOR system privilege can start or stop databases.

- **all –** All users can start or stop databases. Not recommended for network servers that can be accessed by remote clients.
- **none –** Starting and stopping databases isn't allowed except when the database server itself is started and stopped.

Default
The default setting is `DBA` for the network database server. Both uppercase and lowercase syntax are allowed.

Applies to
All operating systems and database servers.

Remarks
This option specifies the level of privilege required for a user to cause a new database file to be loaded by the database server, or to stop a database on a running database server.

When the option is set to DBA, the client application must use an existing connection to another database running on the same server to start and stop databases. You cannot start a database that is not already running by using the DatabaseFile connection parameter.

You can obtain the setting of the -gd option using the StartDBPermission server property:

```
SELECT PROPERTY ( 'StartDBPermission' );
```

The privileges for stopping a database server are specified by the -gk option.

Example

The following steps illustrate how to use the -gd option for the network database server.

1. Start the network database server:
   ```
   iqsrv16 -su mypwd -gd DBA -n my_server
   ```
2. Connect to the utility database from Interactive SQL:
   ```
   dbisql -c "UID=DBA;PWD=mypwd;DBN=utility_db"
   ```
3. Start a database:
   ```
   START DATABASE 'demo.db';
   ```
4. Connect to the database:
   ```
   CONNECT USING 'DBN=demo;UID=DBA;PWD=sql';
   ```

-gk iqsrv16 database server option

Sets the privileges required to stop the database server.

Syntax

```
iqsrv16 -gk { DBA | all | none } ...
```

Allowed values

- **DBA –** Only users with the SERVER OPERATOR system privilege can stop the database server. This is the default for the network server.
- **all –** No privileges are required to shut down the database server.
- **none –** The database server cannot be stopped.

Applies to
All operating systems and database servers.

Remarks
The -gd database server option applies to the dbstop utility as well as to the following statements:

- ALTER DATABASE *dbname* FORCE START statement.
- STOP DATABASE statement

-gl iqsrv16 Server Option

Set the permission required to load data using **LOAD TABLE**.

Syntax
-gl level

Remarks
The **LOAD TABLE** statement reads files from the database server machine. To control access to the file system using these statements, the **-gl** command-line switch allows you to control the level of database permission that is required to use these statements. *level* is either:

- DBA – only users with the LOAD ANY TABLE, ALTER ANY TABLE or ALTER ANY OBJECT system privilege can load data.
- ALL – all users can load data.
- NONE – data cannot be loaded.

You can use either uppercase and lowercase syntax for the options.

The default settings are **all** for servers started with **start_iq** and **dba** for other servers. SAP Sybase recommends that, for consistency with earlier versions, you use the **all** value on all systems. The **all** setting is used in the `iqdemo.cfg` and `default.cfg` configuration files.

-gu iqsrv16 database server option

Sets the privilege required for executing database file administration statements such as for creating or dropping databases.

Syntax

```
iqsrv16 -gu { all | none | DBA | utility_db } ...
```

Allowed values

-gu option	Effect	Applies to
`all`	*This option is deprecated.* Anyone can execute file administration statements.	Any database including utility database
`none`	Executing file administration statements is not allowed.	Any database including utility database
`DBA`	Only users with the SERVER OPERATOR system privilege can execute file administration statements	Any database including utility database
`utility_db`	Only the users who can connect to the utility database can execute file administration statements	Only the utility database

Default
DBA

Applies to
All operating systems and database servers.

Remarks
Restricts the users who can execute the following database file administration statements:

- ALTER DATABASE dbfile ALTER TRANSACTION LOG
- CREATE DATABASE statement
- CREATE DECRYPTED DATABASE statement
- CREATE DECRYPTED FILE statement
- CREATE ENCRYPTED DATABASE statement
- CREATE ENCRYPTED FILE statement
- DROP DATABASE statement
- RESTORE DATABASE statement.

When utility_db is specified, these statements can only be run from the utility database. When DBA is specified, these statements can only be run by a user with the SERVER OPERATOR system privilege. When none is specified, no user can execute these statements.

Examples

To prevent the use of the file administration statements, start the database server using the none privilege level of the -gu option. The following command starts a database server and names it

TestSrv. It loads the `mytestdb.db` database, but prevents anyone from using that server to create or delete a database, or execute any other file administration statement regardless of their resource creation rights, or whether they can load and connect to the utility database.

```
iqsrv16 -n TestSrv -gu none c:\mytestdb.db
```

To permit only the users knowing the utility database password to execute file administration statements, start the server by running the following command.

```
iqsrv16 -n TestSrv -su secret -gu utility_db
```

The following command starts Interactive SQL as a client application, connects to the server named TestSrv, loads the utility database, and connects the user.

```
dbisql -c
"UID=DBA;PWD=secret;DBN=utility_db;Host=host1;Server=TestSrv"
```

Having executed the above command successfully, the user connects to the utility database, and can execute file administration statements.

Maximum Catalog Page Size

The database server cache is arranged in pages, which are fixed-size areas of memory. Because the server uses a single cache for the catalog store until it is shut down, all catalog pages must have the same size.

A catalog file is also arranged in pages, of size 4096, 8192, 16384, or 32768 bytes. Every database page must fit into a cache page.

Use the **-gp** option to set the catalog page size explicitly. By setting **-gp** to the maximum size, 32768, you maximize the number of columns per table that SAP Sybase IQ can support.

By default, the server page size is the same as the largest page size of the databases on the command line. The **-gp** option overrides this default. Once the server starts, you cannot load a database with a larger catalog page size than the server. Unless you specify **-gp**, you cannot load a database file with a catalog page size larger than the databases started on the command line.

If you use larger page sizes, increase your cache size. A cache of the same size accommodates only a fraction of the number of the larger pages, leaving less flexibility in arranging the space.

Note: The **-gp** option and the page sizes listed here apply only to the catalog store. You set the page size for the IQ store in the **IQ PAGE SIZE** parameter of the **CREATE DATABASE** command.

-gp iqsrv16 database server option

Sets the maximum allowed database page size.

Syntax

```
iqsrv16 -gp { 2048 | 4096 | 8192 | 16384 | 32768 } ...
```

Default
4096 (if a database server is started with no databases loaded)

Applies to
All operating systems and database servers.

Remarks
Database files with a page size larger than the page size of the server cannot be loaded. This option explicitly sets the page size of the server, in bytes.

By default, the server page size is the same as the largest page size of the databases on the command line.

On all platforms, if you do not use this option and start a server with no databases loaded, the default value is 4096.

Client/Server Environment Options

The **-x** (communication protocol), **-tl** (network connection timeout), and **-ti** (client connection timeout) options can help you set up your client/server environment.

Communications Protocols

Any communications between a client application and a database server require a communications protocol. SAP Sybase IQ supports a set of communications protocols for communications across networks and for same-machine communications.

The database server supports these protocols:

- *Shared memory* is used for same-machine communications, and is loaded by default.
- *TCP/IP* is supported on all platforms.
- *Named Pipes* is supported on Windows 2000/2003/XP only. Named Pipes is provided for same machine communications to and from Windows client applications using ODBC or Embedded SQL, but is not generally recommended for this purpose. Named Pipes is not used for network communications.

Server Protocols Switch
By default, the database server starts up all available protocols. You can limit the protocols available to a database server by using the **–x** command line switch. On the client side, you can control many of the same options using the **CommLinks** connection parameter.

To start a server using the TCP/IP protocol, use:

```
start_iq -x "tcpip" -n myserver
```

The quotes are not strictly required in this example, but are needed if there are spaces in any of the arguments to **-x**. If you omit this switch and you are using TCP/IP, or if you do not specify a port number, the default port, 2638, is used.

You can add parameters to tune the behavior of the server for each protocol. For example, the following command line instructs the server to use two network cards, one with a specified port number. This command must be entered all on one line, even though it appears on multiple lines here.

```
start_iq
 -x "tcpip(MyIP=192.75.209.12:2367,192.75.209.32)"
   path\iqdemo.db
```

-x iqsrv16 Server Option

Specifies server-side network communications protocols.

Syntax

-x *list*

Remarks

list is a comma-separated list of **tcpip** or **namedpipes** settings. For example:

```
-x tcpip,ipx
```

allows only TCP/IP and IPX communications.

The default is to try all settings supported by the database server on your operating system.

For some protocols, you can provide additional parameters, in this format:

```
-x tcpip(PARM1=value1;PARM2=value2;...)
```

For UNIX, quotation marks are required if more than one parameter is supplied:

```
-x "tcpip(PARM1=value1;PARM2=value2;...)"
```

Switches That Limit Inactive Connections

The server options **-tl** and **-ti** determine when SAP Sybase IQ closes user connections.

Default Network Timeout Switch

A liveness packet is periodically sent across a client/server TCP/IP communications protocol to confirm that a connection is intact. If the server runs for a liveness timeout period (default 2 minutes) without detecting a liveness packet, the communication is severed. The server drops any connections associated with that client. There is no warning. All activity that falls within any open transaction is rolled back.

The **-tl** switch on the server sets the liveness timeout, in seconds, for all clients that do not specify a **-tl** switch when they connect. Liveness packets are sent at an interval of the (liveness timeout)/4.

You may want to set a higher value for this switch at the server level. Many users, especially those who have used earlier versions of SAP Sybase IQ, do not expect to be disconnected after only 2 minutes of inactivity.

Try setting the liveness timeout to 300, together with the recommended value for **–ti**. Set this switch as follows:

```
-tl 300
```

If this value does not work well, try **-tl 1200**, which sets the liveness timeout to 20 minutes.

Note: Users who are running a client and server on the same machine do not experience a liveness timeout.

Default Client Timeout Switch

SAP Sybase IQ disconnects client connections that have not submitted a request for the number of minutes you specify with the **-ti** switch. By disconnecting inactive connections, this option frees any locks those connections hold. The **start_iq** default is 4400 (about 72 hours), which lets you start long runs at the beginning of a weekend, for example, and ensure that any interim results are not rolled back.

-ti iqsrv16 database server option

Disconnects inactive connections.

Syntax

```
iqsrv16 -ti minutes ...
```

Default

240 (4 hours)

Applies to

All operating systems and database servers.

Remarks

Disconnects connections that haven't submitted a request for the specified number of *minutes*. The maximum value is 32767. A client computer in the middle of a database transaction holds locks until the transaction is ended or the connection is disconnected. The -ti option is provided to disconnect inactive connections, freeing their locks.

The -ti option is very useful when used in conjunction with iqsrv16 since most connections will be over network links (TCP).

Setting the value to zero disables checking of inactive connections, so that no connections are disconnected. If the Idle connection parameter is not used, then the idle timeout value for TCP/IP connections is controlled by the -ti database server option. If both the -ti database server option and the Idle connection parameter are specified, then the idle timeout value is controlled by the connection parameter.

-tl iqsrv16 database server option

Sets the period at which to send liveness packets.

Syntax

```
iqsrv16 -tl seconds ...
```

Applies to

All database servers using TCP/IP.

Remarks

A liveness packet is sent periodically across a client/server TCP/IP communications protocol to confirm that a connection is intact. If the server runs for a LivenessTimeout period (default 2 minutes) without detecting a liveness packet on a connection, the communication is severed, and the server drops the connection associated with that client. Unix non-threaded clients and TDS connections do not do liveness checking.

The -tl option on the server sets the LivenessTimeout value for all clients that do not specify a liveness period.

Liveness packets are sent when a connection hasn't sent any packets for between one third and two thirds of the LivenessTimeout value.

When there are more than 200 connections, the server automatically calculates a higher LivenessTimeout value based on the stated LivenessTimeout value, so the server can handle a large number of connections more efficiently. Liveness packets are sent between one third and two thirds of the LivenessTimeout on each idle connection. Large numbers of liveness packets aren't sent at the same time. Liveness packets that take a long time to send could be sent after two thirds of the LivenessTimeout. A warning appears in the database server message log if the liveness sends take a long time. If this warning occurs, consider increasing the LivenessTimeout value.

Although it isn't generally recommended, you can disable liveness by specifying the following:

```
iqsrv16 -tl 0 -n my_server
```

Rather than disabling the LivenessTimeout option, consider increasing the value to one hour as follows:

```
iqsrv16 -tl 3600 -n my_server
```

Forced Recovery Mode Options

After a failure, restart your server with your usual startup options. On rare occasions, you may need to supply startup options to force recovery or to recover leaked storage.

-iqfrec iqsrv16 Server Option

Open database in forced recovery mode.

Syntax
-iqfrec *dbname*

Starting a Server from Interactive SQL

If you are already connected to a running database server, you can start a new server from Interactive SQL.

The following Interactive SQL command, entered on one line, starts a database server, names it `jill_newserv`, and specifies the network connection, number of connections, and catalog page size:

```
START ENGINE AS jill_newserv
STARTLINE 'start_iq -x tcpip(port=5678) -gm 10 -gp 4096'
```

Use the **START ENGINE** command to start a named server from Interactive SQL.

This method is not recommended for most situations. If you use it, be sure you are starting the server on the system you intend, that you include appropriate server parameters in the command, and that environment variables are set appropriately on the system where the server will start.

Shared Memory Conflicts

In a production environment, it would be unusual to have more than one server running on the same system, and SAP Sybase strongly recommends against it. In a development environment, however, this situation can occur.

If you run more than one server or client on the same UNIX machine, and shared memory is enabled, you must take certain precautions to prevent users from connecting to the wrong server.

When attempting to start a server, you may see:

```
DBSPAWN ERROR -96 -- database engine already running
```

This error indicates that the startup process is finding the shared memory segment of a server started earlier, and cannot create a shared memory segment. This error may occur when either a SAP Sybase IQ or SQL Anywhere server is running. (Interactive SQL also connects to an earlier server if its shared memory port is visible, even if you intended for it to connect to a server started later.) You can avoid the error if you run only one server per system, either SAP Sybase IQ or SQL Anywhere.

Ways to Avoid Shared Memory Conflicts

There are several ways to avoid conflicts when using shared memory.

- Create a temporary directory that is dedicated to each server. Make sure that each client uses the same temporary directory as its server by setting the IQTMP16 environment variable explicitly on both systems.
- For each server, create a datasource name in the `.odbc.ini` file (on UNIX) and provide detailed connection information.
- Use connection strings that specify explicit parameters instead of relying on defaults.
- Confirm connections by issuing:

```
SELECT "database name is" = db_name(), "servername_is" =
@@servername
```

If you run multiple servers per system, make sure that each server has a unique:

- Name, specified with the **-n** parameter on startup.
- Port number, specified with the **-x** parameter.

Server Activity Logs

Using commands appropriate for your platform, you can direct SAP Sybase IQ to capture server activity in a log file.

Server Startup Messages

When you start an SAP Sybase IQ server, a series of messages appears in the server log window. The exact set of messages you see depends on your platform and licensed options. This example is from an AIX system:

```
Starting server myserver_iqdemo on myserver at port 3658 (09/06
17:25:23)

Run Directory       : /myserver/users/sybase/iqdemo_160_sep05
Server Executable   : /myserver/users/sybase/160_sep05/IQ-16_0/
bin64/iqsrv16
Server Output Log   : /myserver/users/sybase/160_sep05/IQ-16_0/
logfiles/iqdemo_3658.0001.srvlog
Server Version      : 16.0.0.6556/Mainline
Open Client Version : N/A
User Parameters     : '@iqdemo.cfg' 'iqdemo.db'
Default Parameters  : -ti 4400 -gn 25

I. 09/06 17:25:26. Sybase IQ
I. 09/06 17:25:26. Version 16.0
I. 09/06 17:25:26. (64bit mode)
I. 09/06 17:25:26. Copyright 1992-2012 by Sybase, Inc. All rights
reserved
I. 09/06 17:25:26. Copyright (c) 2001-2012, Sybase, Inc.
I. 09/06 17:25:26. Portions copyright (c) 1988-2011, iAnywhere
Solutions, Inc. All rights reserved.
```

```
I. 09/06 17:25:26. Use of this software is governed by the Sybase
License Agreement.
I. 09/06 17:25:26. Refer to http://www.sybase.com/softwarelicenses.
I. 09/06 17:25:26.
I. 09/06 17:25:26. Processors detected: 4 (containing 16 logical
processors)
I. 09/06 17:25:26. Maximum number of physical processors the server
will use: 4
I. 09/06 17:25:26. Running SunOS 5.10 Generic_144489-12 on X86_64
I. 09/06 17:25:26. Server built for X86_64 processor architecture
I. 09/06 17:25:26. 49152K of memory used for caching
I. 09/06 17:25:26. Minimum cache size: 49152K, maximum cache size:
262144K
I. 09/06 17:25:26. Using a maximum page size of 4096 bytes
I. 09/06 17:25:27. Starting database "iqdemo" (/myserver/users/
sybase/iqdemo_160_sep05/iqdemo.db) at Tue Sep 06 2012 17:25
I. 09/06 17:25:27. Transaction log: iqdemo.log
I. 09/06 17:25:27. Starting checkpoint of "iqdemo" (iqdemo.db) at Tue
Sep 06 2012 17:25
I. 09/06 17:25:27. Finished checkpoint of "iqdemo" (iqdemo.db) at Tue
Sep 06 2012 17:25
I. 09/06 17:25:28. Database "iqdemo" (iqdemo.db) started at Tue Sep
06 2012 17:25
I. 09/06 17:25:28. IQ Server iqdemo_3658.
I. 09/06 17:25:28. Database server started at Tue Sep 06 2012 17:25
I. 09/06 17:25:28. Trying to start SharedMemory link ...
I. 09/06 17:25:28.     SharedMemory link started successfully
I. 09/06 17:25:28. Trying to start TCPIP link ...
I. 09/06 17:25:28. Starting on port 3658
I. 09/06 17:25:33.     TCPIP link started successfully
I. 09/06 17:25:33. Now accepting requests
I. 09/06 17:25:45. Database server shutdown due to HUP signal
I. 09/06 17:25:45. TCPIP listener on IP address (::):3658 is exiting
I. 09/06 17:25:45. TCPIP listener on IP address 0.0.0.0:3658 is
exiting
I. 09/06 17:25:45. Starting checkpoint of "iqdemo" (iqdemo.db) at Tue
Sep 06 2012 17:25
I. 09/06 17:25:45. Finished checkpoint of "iqdemo" (iqdemo.db) at Tue
Sep 06 2012 17:25
I. 09/06 17:25:46. Database server stopped at Tue Sep 06 2012 17:25
```

The start_iq Log File

When you start a server with the **start_iq** utility, server activity is logged in an ASCII text file placed in the directory defined by $IQLOGDIR16. This file contains the standard output from the server and the server status.

The log file name has this format:

```
your_server_name.nnnn.srvlog
```

Each time you start the server, the number is incremented. For example, your directory may look like this:

```
demo.0001.srvlog   demo.0002.srvlog
testdemo.0001.srvlog
```

For information about your most recent session, choose the log with the largest number for the desired server. Issue a **tail –f** command to view the log contents. For example:

```
% tail -f demo.0002.srvlog
```

If you do not define `$IQLOGDIR16` directory, then on UNIX, the log is written to `$IQDIR16/logfiles/` directory, and on Windows to the `$IQLOGDIR16` directory defined by the SAP Sybase IQ installation.

When you run **start_iq**, specify the **-z** option to enhance the log file with additional information about connections, which may help new users or those troubleshooting connection problems.

On UNIX systems, there are two ways to check if a server is running.

- Log in to the machine where the server was started, and issue:

  ```
  ps -eaf | grep iqsrv
  ```

  This output differs slightly across UNIX platforms. For IBM AIX, the columns are:

  ```
  UID   PID  PPID   C    STIME TTY          TIME CMD
  ```

  For example:

  ```
  jones  422034      1   0 17:47:36      -  0:04
  /ibm64srv/users/sybase/iq160/IQ-16_0/bin64/
  iqsrv16
  @iqdemo.cfg iqdemo.db -ti 4400 -gn 25 -o
  /ibm64srv/users/sybase/iq160/IQ-
  16_0/logfiles/ibm64srv_iqdemo.0003.srvlog -hn 7
  ```

- Use the **stop_iq** utility, described in the following section, which displays all SAP Sybase IQ processes that are running.

On Windows systems, look in the system tray for one or more SAP Sybase IQ icons. Place the cursor over each icon and read the server name.

Naming the Server Log File

The server log name defaults to `server.nnnn.srvlog`.
Use the **-o** switch on the **start_iq** command to change the server log file name.

For example, to save output to a file named `results` in the directory where the server was started, start the server using:

```
start_iq -n imyserver -o results
```

You can also use the **-o** switch to specify the full path to the log file.

UNIX Log Files

On UNIX platforms, an additional log file captures operating system output, including stdout and stderr output.

The file name has this format:

```
your_server_name.####.stderr
```

For unexpected exceptions, SAP Sybase IQ writes a stack trace file. On UNIX systems, the name of the file that contains stack trace information has this format:

```
stktrc-YYYYMMDD-HHNNSS_#.iq
```

Database Server Shutdown

System administrators must know how to stop the SAP Sybase IQ database server, and when it is necessary, how to control who can stop it, and how to stop the server when you shut down the operating system.

When to Stop and Restart the Server

In some situations, the server may need to be stopped and restarted.

For example:

- To install a new version of SAP Sybase IQ
- To reset some server command line options
- To cause a small number of server-wide database options to take effect
- Before closing the operating system session

See also

- *Scope and Duration of Database Options* on page 411

Stopping Servers

Different ways to stop a server.

SERVER OPERATOR system privilege is the default permission level required to stop a server, but you can use the **-gk** startup option to change the default to ALL or NONE. Setting the permission level to ALL lets all users stop the server; setting the permission level to NONE means that no one can stop the server. In a production environment, only the Server Operator is allowed to stop the database server.

Do not shut down a server that is still connected to a client, or you will lose uncommitted transactions. Disconnect or close all the clients before shutdown.

Server Shutdown Commands on UNIX

Several commands shut down the database server.

Command	Description
stop_iq	Shuts down a server completely without regard for users, connections, or load process status. Syntax: `stop_iq [ options ]`
dbstop	Stops a server or database. Additional options let you stop a server, even if there are active connections. Syntax: `dbstop [ options ] server-name` To use **dbstop** to stop a server, you must specify a **server-name**, as well as any connection parameters you specified when you started the server.
iqsrv16	Stops a server when typed in the window where the server was started, if you have not redirected input to another device. `q`
STOP ENGINE	**STOP ENGINE** is a SQL statement that stops a database server. Syntax: `STOP ENGINE engine-name [ UNCONDITIONALLY ]` The UNCONDITIONALLY argument stops a database server, even if there are client connections to the server.

Server Shutdown Methods on Windows

There are several methods to shut down or stop a server.

To stop the server from..	Do this...
Command line	Run the **stop_iq** or **dbstop** command.
Desktop	Right-click the **server icon** in the system tray, and choose **Shutdown server name**.
Service Manager	If you started the server as a Windows service: 1. On the **Control Panel**, choose **Administrative Tools** > **Services**. 2. Choose the SAP Sybase IQ service, then click **Stop the service**.

To stop the server from..	Do this...
Interactive SQL	Run the **stop engine** command to stop a named database server.

Stopping Servers in Cron or At Jobs on UNIX

To use **stop_iq** in a **cron** or **at** job, specify the full path name to the **stop_iq** executable and the appropriate **–stop** option.

Setting **–stop one** shuts down a single server, when exactly one running server was started by the user ID that starts the **cron** or **at** job. This prevents accidentally shutting down the wrong server if several are running:

```
stop_iq -stop one
```

Setting **–stop all** shuts down all servers that were started by the user ID that starts the **cron** or **at** job:

```
stop_iq -stop all
```

Example — Stop a Server Using stop_iq

The following example uses the **stop_iq** utility in a UNIX operating system command line to shut down an SAP Sybase IQ server and close all user connections to it.

When you issue the **stop_iq** command, SAP Sybase IQ lists all servers owned by other users, followed by the servers you own. It then asks if you want to stop your server. For example:

```
% stop_iq
```

```
Checking system for IQ 16 Servers ...
The following 2 server(s) are owned by other users.
##     Owner   PID   Started   CPU_Time  Additional Information
--     -----  -----  --------  --------  ----------------------
      handari  19895  15:43:44   183:38
start_iq @iqdemo.cfg iqdemo.db -gn 105 -o /server1/users/surya/
IQ-16_0/logfiles/surya_ibm2.001.srvlog -hn 8      pamela    409802
18:05:02    0:05  SVR:ibm1_iqdemo2 DB:iqdemo
PORT:2678/ibm1/users/sybase/iq160/IQ-16_0/bin64/iqsrv16 @iqdemo.cfg
iqdemo.db -ti 4400 -gn 25 -o /ibm1/users/sybase/iq160/IQ
16_0/logfiles/ibm64qa_iq
The following 1 server(s) are owned by 'kermit'
##     Owner    PID   Started   CPU_Time  Additional Information
-- ---------   -----  --------  --------  ----------------------
1:     kermit  422034  15:11:37     0:07  SVR:myserver_iqdemo
DB:iqdemo PORT:2638 /myserver/users/sybase/iq160/IQ-16_0/bin64/
iqsrv16
@iqdemo.cfg iqdemo.db -ti 4400 -gn 25 -o /myserver/users/sybase/
iq160/IQ-
16_0/logfiles/myserver_iq
start_iq -c 32m -gd all -gm 10 -gn 25 -gp 4096 -ti 4400 -tl 300
@iqdemo.cfg

--
```

```
    Please note that 'stop_iq' will shut down a server completely
    without regard for users connections or load processes status.
    For more control, use the 'dbstop' utility, which has options
    that control stopping servers based on active connections.

Do you want to stop the server displayed above <Y/N>?
```

To shut down the server, type `Y` (yes). You see messages similar to:

```
Shutting down server (422034) ...
Checkpointing server (422034) ...
Server shutdown.
```

To leave the server running, type `N` (no). You return to the system prompt and the server does not shut down.

If no running servers were started by your user ID, SAP Sybase IQ displays information about servers run by other users, then a message similar to:

```
There are no servers owned by 'kermit'
```

Example — Stopping a Server From Interactive SQL
The following example stops a server from Interactive SQL:

```
STOP ENGINE Ottawa UNCONDITIONALLY
```

The optional keyword **UNCONDITIONALLY** specifies that the database server will be stopped even if there are connections to it.

Note: You can stop a server from Interactive SQL if you are connected as DBA to one of the databases running on that server (including the `utility_db` database), or if the server was started with the **-gk ALL** option.

Permissions Required to Stop the Server

When you start a server, you can use the **-gk** option to set the level of permissions required for users to stop the server with **DBSTOP** or **STOP ENGINE**.

The default level of permissions required is **DBA**, which requires the SERVER OPERATOR system privilege, but you can also set the value to **ALL** or **NONE**. If you set it to **NONE**, even the a user with SERVER OPERATOR system privileges cannot execute **STOP ENGINE**. In a production environment, only the DBA should be allowed to stop the database server.

Running **stop_iq** at the UNIX command line, or Shutdown on Windows platforms, still allows you to stop the server and databases on the machine where the server was started.

Operating System Session Shutdown

Always stop the database server explicitly before closing the operating system session.

If you close an operating system session where a database server is running, or if you use an operating system command (other than **stop_iq**) to stop the database server, the server shuts down, but not cleanly. Next time the database is loaded, recovery happens automatically.

Note: An example of a command that does not stop a server cleanly is stopping the process in the Windows Task Manager Processes window.

Starting and Stopping Databases

You can start databases when you start the server, or after the server is running.

Run only one database per server, especially in a production environment.

There are several ways to start a database on a running server:

- To start a database from Interactive SQL or Embedded SQL, use the **START DATABASE** statement.
- To start and connect to a database from Interactive SQL, use a data source that specifies the database file.
- To start and connect to a database when you start Interactive SQL from a system command prompt, include the connection parameter **"DBF=db-file"**.
- To start an embedded database while connected to a server, connect to a database using a DBF parameter that specifies a database file for a new connection. The database file is loaded onto the current server.

Database Startup Guidelines

There are several issues to consider regarding database startup.

File Access

For a database to start, all files of IQ_SYSTEM_MAIN, all files of IQ_SYSTEM_TEMP, and the catalog file SYSTEM must be available. You can start a database while skipping dbspaces that cannot be fully opened. If any writeable files of IQ main store dbspaces other than IQ_SYSTEM_MAIN or any catalog dbspace files other than SYSTEM cannot be opened on server startup, SAP Sybase IQ logs an error and marks the dbspace dynamically offline (marked offline in memory, as opposed to marking it offline in the catalog). If there are any files of IQ_SYSTEM_TEMP that cannot be opened, the database does not start unless you use the **-iqnotemp** startup parameter.

SAP Sybase IQ checks the consistency of the commit_id in each dbspace file header against the value in the system tables `ISYSDBFILE` and `ISYSIQDBSPACE` and marks any file or dbspace that does not match offline as above.

A dbspace that has been marked offline at start time may be brought online via the **ALTER DBSPACE ONLINE** statement, assuming that the problem has been corrected and the dbspace can be opened. To correct path problems, you can correct the path of the dbspace file using **ALTER DBSPACE** *dbspace name* **ALTER FILE** *logical filename* **RENAME PATH** *new pathname*.

A table object that resides in an offline dbspace is unavailable. Any DDL or DML request except **ALTER DBSPACE ONLINE**, to any table object in an offline dbspace generates an error.

After you make a dbspace offline, there may still be data pages in the buffer cache. In the case of a very small table, the entire table may be in memory in the buffer cache and temporarily available, even if the dbspace is offline.

Page Size Limitations

The server holds database information in memory using fixed-size pages. Once a server has been started, you cannot load a database that has a catalog or IQ page size larger than the server. For this reason, always set the catalog page size to its maximum value, 32768 bytes, with the **-gp** switch.

Permission Limitations

The **-gd** server command line option determines the permission level required to start databases. By default, this option is set to **DBA**, so that only users with the SERVER OPERATOR system privilege can start IQ databases. However, you can also set this option to **ALL** or **NONE**. **ALL** means that all users can start a database. **NONE** means that no users, including the user with the SERVER OPERATOR system privilege, can start a database.

Stopping Databases

Different ways to stop a database.

- Disconnect from a database started by a connection string. The database stops automatically when the last user disconnects from it, unless you explicitly set the **AUTOSTOP** connection parameter to NO.
- From Interactive SQL or Embedded SQL, use the **STOP DATABASE** statement.

See *STOP DATABASE Statement [Interactive SQL]* in *Reference: Statements and Options*.

See also

- *STOP DATABASE Statement [Interactive SQL]* on page 498

Starting the iqdemo Database

Use the script provided at installation to create the `iqdemo` database and the configuration file to start it.

This configuration file, called `iqdemo.cfg`, contains all the parameters necessary to start the demo database. See *Quick Start*

Connect to SAP Sybase IQ Servers and Databases

SAP Sybase IQ runs in a client/server environment, in which many users can connect to a database server across a network.

You may be able to connect to more than one database server; specify connection options that clearly identify servers.

Note: You can connect from Interactive SQL on a Windows or Linux client to SAP Sybase IQ on a UNIX server.

Client applications can connect to databases from ODBC, OLE DB, Embedded SQL applications, and Interactive SQL.

Any client application that uses a database must establish a *connection* to that database before any work can be done. The connection forms a channel through which all activity from the client application takes place. For example, your user ID determines permissions to carry out actions on the database—and the database server has your user ID because it is part of the request to establish a connection.

Roadmap for Connections

The roadmap shows which topics address specific connection needs.

To ...	Click the related topic link at the bottom of this page ...
Connect using a client application	*Supported Connection Interfaces*
Customize a connection string	*Connection and Communication Parameters*
Create data sources	*ODBC Data Sources*
Diagnose network connection issues	*Troubleshooting Network Communications*

See also

- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63

- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106
- *Appendix: Connection and Communication Parameters Reference* on page 363
- *Troubleshooting Network Communications* on page 339

Ways to Connect

Applications connect to SAP Sybase IQ databases in various ways.

To query the `iqdemo` database on a local server, you may only need the connection parameters shown.

Most IQ application environments require a more complex set of connection parameters.

See also

- *Roadmap for Connections* on page 41
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100

- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Connecting to the Demo Database from Interactive SQL

Many examples and exercises throughout the documentation start by connecting to the demo database from Interactive SQL, also called **dbisql**.

1. Select **Programs > Sybase > Sybase IQ 16.0 > Interactive SQL**.
2. On the Identification tab, type `DBA` and `sql` for the User and Password.
3. On the Database tab, choose **Find**.
4. Select your iqdemo server from the Find Servers screen and click **OK**.

Use the same process to connect to any database server that is already running. You can also specify a nondefault character set and language.

How Database Status Affects Local Connections

The simplest connection scenario is when the database you want to connect to resides on your own machine.

If this is the case:

- Is the database already running on a server? If so, you can specify fewer parameters in the Connect dialog. If not, identify the database file so that Interactive SQL can start it.
- Are there multiple databases running on your machine? If so, identify the database to which you want Interactive SQL to connect. If there is only one database, Interactive SQL connects to it, and you do not need to specify it in the Connect dialog.

Connecting to a Running Database on a Local Server

When the database is already running on a local server, you may specify fewer Connect dialog parameters than usual.

1. Start Interactive SQL and open the Connect dialog, if it does not appear automatically.
2. On the **Identification** tab, enter a user ID and a password.
3. Do one of the following:
   - If the server contains only one database, click **OK** to connect to it.
   - If the server contains multiple databases, click the **Database** tab and specify a database name. This is usually the database file name, without the path or extension.

Note: If the database is already loaded (started) on the server, you need only provide a database name for a successful connection. The database file is unnecessary.

Connecting to a Database from Interactive SQL on UNIX

You do not need to specify the host and port for a database on your local machine.

1. At a system command prompt, start the server and the database:

   ```
   start_iq dbname
   ```

2. Start Interactive SQL:

   ```
   dbisql -c "uid=userID;pwd=password" -host hostname -port portnum -
   n servername dbfilename.db
   ```

   The **-c** parameter specifies connection parameters.

   For example, to connect to the demo database on remote host `fiona`, enter:

   ```
   dbisql -c "uid=DBA;pwd=sql" -host fiona -port 1870 -n fiona_iqdemo
   $IQDIR16/demo/iqdemo.db
   ```

Connecting from a UNIX System

Connect to a running database from the command line on a UNIX system.

Prerequisites

Make sure that your PATH and other environment variables are correctly set.

Task

1. To ensure that the demo database is loaded on a running server, at the UNIX prompt, enter:

   ```
   ps -eaf | grep iqdemo
   ```

   ```
   cd $IQDIR16/demo
   ```

2. To start the demo database, enter:

   ```
   start_iq @iqdemo.cfg iqdemo.db
   ```

3. Start Interactive SQL:

   ```
   dbisql -c "uid=DBA;pwd=sql;eng=servername;links=tcpip"
   ```

   Replace *servername* with the same server name that was supplied in the **start_iq** command to start the server.

   Note: If you prefer the older utility Interactive SQL Classic to the Java-based version, enter **dbisqlc** instead of **dbisql**. However, **dbisqlc** does not contain all the features of **dbisql**.

   The **–c** parameter specifies connection parameters. You can also specify these parameters in a data source, as described later in this chapter.

   Note: The `links=tcpip` (or `CommLinks=tcpip`) parameter is only required if you use TCP/IP to connect to the database. If you use the shared memory port to connect to a

local database you can omit the `links` parameter; however, it is always safer—and required on some platforms—to include complete network parameters.

To connect to a database on a remote host, you must add the host name and port number. For example:

```
dbisql -c "uid=DBA;pwd=sql;eng=SERV1_iqdemo;
links=tcpip(port=1234)"
```

Connecting from a Windows System

Connect to a running database from the Start menu or command prompt.

1. Choose **Programs > Sybase > Sybase IQ 16.0 > Interactive SQL**, or at the Windows command prompt, enter:

   ```
   dbisql
   ```

   You can include the **-c** parameter to specify connection parameters in the **dbisql** command. If you omit these parameters, the Interactive SQL connect dialog appears.
2. In the Connect dialog, enter your user name and password.

   For example, for the `iqdemo` database, enter `DBA` and `sql`, the default user and password combination for SAP Sybase IQ databases when they are created.
3. Click the **Database** tab and type the server name that was used to start the server (for example, "*hostname*_iqdemo" for the `iqdemo` database). This name must be unique on your local area network.

   For remote servers, specify the server as *host name* and *port number* on the **Network** tab.

   The default port number is 2638, but if the server was started with a different number, use that instead. You can find the port number by running **Sybase IQ 16.0 > ODBC Administrator 32-bit** or **Sybase IQ 16.0 > ODBC Administrator 64-bit**. Select the **User Data Sources on the User DSN** tab, then click **Configure**. Enter `dblocate` at the command prompt.

   This procedure connects you to the first database started on this server. If more than one database is running, you may need to click Browse to select the database you want.
4. Click **OK** to connect to the database.

   If you see the **Connect** dialog or an error message about missing information, enter the **-host** and **-port** or other missing information in the **Advanced** tab. If your database is on a remote server, enter the **-host** and **-port** parameters on separate lines, for example:

   ```
   -host fiona
   -port 1870
   ```
5. After you connect to the database, the Interactive SQL window appears. The database name, user ID, and server name for the connection appear on its title bar.

If you connect using Interactive SQL Classic, you see `Connected to database` in the Statistics window, along with a message that shows the collation used by the database.

Connections to Embedded Databases

An *embedded database*, designed for use by a single application, runs on the same machine as the application and is largely hidden from the application user.

When an application uses an embedded database, the database is generally not running when the application connects. You can start the database using the connection string, and by specifying the database file in the DatabaseFile (DBF) parameter of the connection string.

Database File Parameter

The DBF parameter specifies which database file to use. The database file automatically loads onto the default server, or starts a server if none is running.

The database unloads when there are no more connections to the database (generally when the application that started the connection disconnects). If the connection started the server, it stops once the database unloads.

Use these connection parameters to load the demo database as an embedded database:

```
dbf=path\iqdemo.db
uid=DBA
pwd=sql
```

where *path* is the name of your SAP Sybase IQ installation directory.

Start Parameter

The following connection parameters show how to customize the startup of the demo database as an embedded database. You may find this useful if you use command-line options, such as the cache size:

```
Start=start_iq -gd all
-gl all -gm 10 -gn 25 -gp 4096 -c 32M
-ti 4400 -tl 300
dbf=path\iqdemo.db
uid=DBA
pwd=sql
```

Example: Connecting from Interactive SQL

In this example, the demo database is an embedded database within Interactive SQL.

Connecting To an Embedded Database From Interactive SQL in Windows

To connect to an embedded database, do the following:

1. Start SAP Sybase IQ with no databases running. You can use either of the following ways:
   - From the Windows Programs menu, choose **Sybase > Sybase IQ 16.0 > Interactive SQL**, or,

- Type `dbisql` at a system command prompt.

When Interactive SQL starts, it is not connected to any database.

2. Type CONNECT in the command window, and press F9 to execute the command. The connection dialog appears.
3. If you have an ODBC data source for your database, select that data source.
4. Enter `DBA` as the user ID and `sql` as the password. Then click the Database tab. Enter the full path of the demo database in the Database File field. For example, if your installation directory is `c:\sybase\IQ-16_0`, enter:

```
c:\sybase\IQ-16_0\iqdemo.db
```

5. Leave all other fields blank, and click OK. SAP Sybase IQ starts up and loads the demo database, and SAP Sybase IQ connects to the database.

TCP/IP protocol

SAP Sybase IQ uses TCP/IP to connect clients to databases running on different computers.

When you use the TCP/IP protocol, you can secure client/server communications using transport-layer security and RSA encryption technology.

UDP

UDP is a transport layer protocol that sits on top of IP. SAP Sybase IQ may use UDP to do initial server name resolution and then use TCP for connection and communication.

UDP packets sent by the database server in response to client broadcasts contain no sensitive information. The data contained in these packets is limited to:

the database server name
the port number
the database server version
the names of databases running on the database server

You can hide database names from UDP broadcast responses by using the -dh database option.

You can specify the -sb server option to disable the UDP listeners.

Using TCP/IP with Windows

The TCP/IP implementation for database servers on all Windows platforms uses Winsock 2.2. Clients on Windows Mobile use the Winsock 1.1 standard.

Client/server communications encryption over TCP/IP

By default, communication packets are not encrypted, which poses a potential security risk. You can secure communications between client applications and the database server over TCP/IP using simple encryption or transport-layer security. Transport-layer security provides server authentication, strong encryption using RSA encryption technology, and other features for protecting data integrity.

IPv6 support in SAP Sybase IQ

On IPv6-enabled computers, the network database server listens by default on all IPv4 and IPv6 addresses. IPv6 is not supported on Windows CE.

Usually no changes are required to the database server start line to use IPv6. In a case where specifying an IP address is required, the database server and the client libraries both accept IPv4 and IPv6 addresses. For example, if a computer has more than one network card enabled, it probably has two IPv4 addresses and two IPv6 addresses. For IPv6 addresses that include a port number, you must enclose the address in either square brackets or parentheses. If you want the database server to listen on only one of the IPv6 addresses, you can specify an address in the following format:

```
iqsrv16 -x tcpip(MyIP=fd77:55f:5a64:52a:202:5445:5245:444f) ...
```

Similarly, if a client application needs to specify the IP address of a server, the connection string or ODBC data source can contain the address, in the following format:

```
...HOST=fd77:55f:5a64::444f;...
```

Each interface is given an interface identifier, which appears at the end of an IPv6 address. For example, if `ipconfig.exe` lists the address `fd77:55f:5a64::444f`, the interface identifier is 7. When specifying an IPv6 address on a Windows platform, the interface identifier should be used. On Unix, you can specify either an interface identifier or an interface name (the interface name is the name of the interface reported by ifconfig). For example, the interface name is eth1 in the following IPv6 address: `fd77:55f:5a64::444f;`. An interface identifier is required when specifying IPv6 addresses on Linux (kernel 2.6.13 and later). This requirement affects values specified by the following:

Example

Suppose `ipconfig.exe` lists two interfaces, one with the identifier 1 and the other with the identifier 2. If you are looking for a database server that is on the network used by interface number 2, you can tell the client library to broadcast only on that interface:

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

The IPv6 link-local multicast address is `ff02::1`.

Firewall connections

There are restrictions on connections when the client application is on one side of a firewall and the database server is on the other side. Firewall software may filter network packets according to network port. Also, it is common to disallow UDP packets from crossing the firewall.

Usually, you can connect through a properly configured firewall using the Host connection parameter and providing the database server address and port. If the database server is using the default port of 2638, the port is not required.

The following connection string fragment connects to a database server named myserver running on a computer at address serverhost using port 2020. No UDP packets are used because the Host connection parameter specifies the TCP/IP address and port.

```
Server=myserver;Host=serverhost:2020
```

Firewalls that only allow certain client ports

If the firewall must be configured to allow only certain client ports, then you must use the CommLinks(LINKS) connection parameter instead of the Host connection parameter. The following TCP/IP protocol options are required when you use the CommLinks connection parameter:

- **Host –** Set this protocol option to the host name on which the database server is running. You can use the short form IP.
- **ServerPort –** If your database server is not using the default port of 2638, you must specify the port it is using. You can use the short form PORT.
- **ClientPort –** Set this protocol option to a range of allowed values for the client application to use. You can use the short form CPORT.
- **DoBroadcast=NONE –** Set this protocol option to prevent UDP from being used when connecting to the server. You can use the short from DOBROAD.

Your firewall must be configured to allow TCP/IP traffic between the SAP Sybase IQ database server's address and all the SAP Sybase IQ clients' addresses. The SAP Sybase IQ database server's address is the IP address of the computer running the SAP Sybase IQ server (the HOST protocol option) and the SAP Sybase IQ database server's IP port number (the ServerPort protocol option, default 2638).

Use a range with more client ports than the maximum number of concurrent connections from each client computer since there is a several minute timeout before a client port can be reused. The range of clients specified in the ClientPort protocol option must match the range allowed by the firewall.

Example

The following connection string fragment restricts the client application to ports 5050 through 5060, and connects to a database server named myeng running on the computer at address myhost using the server port 2020. No UDP broadcast is performed because the DoBroadcast protocol option is set to NONE.

```
Server=myeng;LINKS=tcpip(ClientPort=5050-5060;HOST=myhost;PORT=2020
;DoBroadcast=NONE)
```

Dial-up network connections (CommLinks connection parameter)

To connect over a dial-up network connection, you must use the CommLinks connection parameter.

On the client side, you should specify the following protocol options:

- **Host –** You should specify the host name or IP address of the database server using the Host (IP) protocol option.
- **DoBroadcast –** If you specify the Host (IP) protocol option, there is no need to do a broadcast search for the database server. For this reason, use direct broadcasting.
- **MyIP –** You should set `MyIP=NONE` on the client side.
- **TIMEOUT –** Set the Timeout (TO) protocol option to increase the time the client waits while searching for a server.

Example

The following is an example of a typical CommLinks (LINKS) connection parameter used to connect over a dialup network connection:

```
LINKS=tcpip(MyIP=NONE;DoBroadcast=DIRECT;HOST=server-ip);
TIMEOUT=15)
```

Connecting Using a Data Source

You can save sets of connection parameters in a data source. ODBC and JDBC using the iAnywhere JDBC driver use data sources, as do Embedded SQL applications like Interactive SQL Classic.

You can create data sources from the ODBC Administrator.

All applications can benefit from using data sources.

The iqdemo data source holds a set of connection parameters, including the database file and a start parameter to start the demo database. The server name in this data source is "*hostname*_iqdemo" where hostname represents your system name.

Connecting using an ODBC Data Source

Connect from Interactive SQL using an ODBC data source.

1. Start Interactive SQL and open the Connect dialog (if it doesn't appear automatically).
2. On the Identification tab (Login tab in Interactive SQL Classic), enter a user ID and password, for example, `DBA` and `sql`.
3. On the lower half of the Identification tab, do one of the following:
   - Select the ODBC Data Source Name option and specify a data source name (equivalent to the DSN connection parameter, which references a data source in the registry). To view a list of data sources, click Browse.
   - Select the ODBC Data Source File option and specify a data source file (equivalent to the FileDSN connection parameter, which references the data source held in a file.). You can search for a file by clicking Browse.

The SAP Sybase IQ Demo data source holds a set of connection parameters, including the database file and a start parameter.

Note: You can also specify the data source name by including the **dsn** connection parameter when you start Interactive SQL:

```
dbisql -c "dsn=Sybase IQ Demo"
```

Default Connection Parameters

You can leave many connection parameters unspecified, and instead use the default behavior to make a connection.

Note: Be extremely cautious about relying on default behavior in production environments, especially if you distribute your application to customers who may install other SAP Sybase IQ or SQL Anywhere applications on their machine.

Default Database Server

If you are connecting to a database on your local server, and more than one database has been started on that server, you need to specify the database you wish to connect to, but you can leave the server as a default:

```
dbn=db_name
uid=user_id
pwd=password
```

Note: Do not use these parameters if more than one local server is running, or you may connect to the wrong server.

Default Database

If more than one server is running, you need to specify which one you wish to connect to. If only one database has been started on that server, you do not need to specify the database name. The following connection string connects to a named server, using the default database:

```
eng=server_name
uid=user_id
pwd=password
```

Connections Without Defaults

The following connection string connects to a named server, using a named database:

```
eng=server_name
dbn=db_name
uid=user_id
pwd=password
```

Connecting from SAP Sybase IQ Utilities

SAP Sybase IQ database utilities that communicate with the server (rather than acting directly on database files) do so using Embedded SQL.

How Database Utilities Obtain Connection Parameter Values

Many of the administration utilities obtain the connection parameter values by:

Using values specified on the command line (if any). For example, the following command takes a backup of the catalog store on the `demo` database, using the user ID `DBA` and the password `sql`:

```
dbbackup -y -x -c
'uid=DBA;pwd=sql;eng=iqdemo;dbn=iqdemo.db;links=tcpip{port=2638}' -
d  '/mydir'
```

1. Using the SQLCONNECT environment variable settings if any command line values are missing. SAP Sybase IQ database utilities do not set this variable automatically. This option provides better password security than other methods. For a description of the SQLCONNECT environment variable, see *Reference: Building Blocks, Tables, and Procedures > File Locations and Installation Settings > Environment Variables*.
2. Prompting you for a user ID and password to connect to the default database on the default server, if parameters are not set in the command line or the SQLCONNECT environment variable.

For a description of command-line options for each database utility, see the *Utility Guide*.

Supported Connection Interfaces

To establish a connection, the client application calls functions in one of the supported interfaces.

SAP Sybase IQ supports the following interfaces:

- ODBC
- OLE DB
- Embedded SQL
- SAP Sybase Open Client — See *SAP Sybase IQ as a Data Server for Client Applications* in *Programming*.
- JDBC — See *JDBC CLI* in *Programming*. Interactive SQL supports the iAnywhere JDBC driver, which is included with SAP Sybase IQ.

Note: The iAnywhere JDBC driver is deprecated.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79

- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Connection Status

Some client tools may not clearly indicate connection status. A failed command is your first indication that the connection does not exist.

A quick way to confirm the connection is by querying the database name.

To display the current database, use this syntax:

```
select db_name()
```

To specify a different database, use this syntax:

```
select db_name([ database_id ])
```

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100

- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

How Connection Parameters Work

When an application connects to a database, it uses a set of *connection parameters* to define the connection.

Connection parameters include information such as the server name, the database name, and a user ID.

A keyword-value pair, of the form *parameter=value*, specifies each connection parameter. For example, you specify the password connection parameter for the default password as follows:

```
Password=sql
```

Connection parameters are assembled into connection strings. In a connection string, a semicolon separates each connection parameter, as follows:

```
ServerName=host_iqdemo;DatabaseName=iqdemo
```

Several connection parameters affect how a server is started. It is recommended that you use the following connection parameters instead of providing the corresponding server options within the StartLine (START) connection parameter:

- EngineName (ENG)
- DatabaseFile (DBF)
- DatabaseName (DBN)

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94

- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Format for Connection Strings

Examples in this document represent connection strings in the following form:

```
parameter1=value1
parameter2=value2
...
```

This is equivalent to the following connection string:

```
parameter1=value1;parameter2=value2
```

You must enter a connection string on a single line, with the parameter settings separated by semicolons.

How Applications Pass Connection Parameters

Connection parameters are passed to the interface library as a *connection string*.

This string consists of a set of parameters, separated by semicolons.

In general, the connection string built up by an application and passed to the interface library does not correspond directly to the way a user enters the information. Instead, a user may fill in a dialog box, or the application may read connection information from an initialization file.

Certain SAP Sybase IQ utilities accept a connection string as the **-c** command-line option and pass the connection string on to the interface library without change. For example, to stop a database named `iqdemo` on the server `myserver`, enter:

```
dbstop -c "uid=DBA;pwd=sql;eng=myserver;dbn=iqdemo"
```

Note: Interactive SQL processes the connection string internally. It does not simply pass on the connection parameters to the interface library. Do not use Interactive SQL to test connection strings from a command prompt.

Connection Parameters in ODBC Data Sources

Many client applications, including application development systems, use the ODBC interface to access SAP Sybase IQ.

When connecting to the database, ODBC applications typically use ODBC data sources. An ODBC data source is a set of connection parameters, stored in the registry or in a file.

For SAP Sybase IQ, ODBC data sources can be used not only by ODBC applications on Windows, but also by other applications:

- SAP Sybase IQ client applications on UNIX can use ODBC data sources, as well as those on Windows operating systems. On UNIX, the data source is stored as a file.
- SAP Sybase IQ client applications using the OLE DB or Embedded SQL interfaces can use ODBC data sources, as well as ODBC applications.
- Interactive SQL can use ODBC data sources.
- JDBC connections using the iAnywhere JDBC driver can use ODBC data sources.

Note: The iAnywhere JDBC driver is deprecated.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Interactive SQL Connections

You must connect to your database in order to manage it with Interactive SQL.

In the Connect dialog, you tell Interactive SQL what database you want to connect to, where it is located, and how you want to connect to it.

The connecting process depends on your situation. For example, if you have a server already running on your machine and this server contains only one database, all you have to do in the

Connect dialog is provide a user ID and a password. Interactive SQL then knows to connect immediately to the database on the running server.

If this running server has more than one database loaded on it, if it is not yet running, or if it is running on another machine, you need to provide more detailed information in the Connect dialog so that Interactive SQL connects to the right database.

This section describes how to access the Connect dialog in Interactive SQL.

Note: To avoid ambiguity, specify connection parameters for Interactive SQL instead of relying on defaults. You can specify connection parameters in a command line, configuration file, or an initialization file such as `.odbc.ini` or `odbc.ini`.

If more than one database is started on a server, for example, you should specify the database name. In a network with subnets, specify the **CommLinks** parameter with protocol options including the host number.

In the `.odbc.ini` file, you must use the long form of each parameter. For example, use DatabaseFile instead of DBF. If your parameters are incomplete or incorrect, you may see an error such as

```
Database name required to start engine
```

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

The Connect Dialog

The Connect dialog lets you define parameters for connecting to a server or database.

The Connect dialog has the following tabs:

- The Identification tab lets you identify yourself to the database and specify a data source.
- The Database tab lets you identify a server and/or database to connect to.
- The Network tab lets you specify either the shared memory or TCP/IP connection protocol, choose a security option, and specify encryption parameters.
- The Advanced Options tab lets you add connection parameters and specify a driver for the connection.

After you connect, the database name appears in the left pane of the main window, under the name of the server where it runs. The user ID for the connection appears in brackets after the database name.

After you connect in Interactive SQL, the connection information, including the database name, your user ID, and the database server, appears on a title bar above the SQL Statements pane.

Opening the Connect dialog (Interactive SQL)

A Connect dialog in Interactive SQL lets you connect to a database.
In Interactive SQL, choose File > New Window or SQL > Connect

Alternatively, you can press F11 to open the Connect dialog.

Once the Connect dialog appears, you must specify the connection parameters you need to connect. For example, connect to the SAP Sybase IQ demo database by specifying `iqdemo.db` as the database file, using the Browse button on the Database tab, and typing `User ID DBA` and `Password sql` on the Identification tab and clicking OK.

If the server is remote, make sure to select “Search network for database servers,” on the Database tab.

Note: When you connect to a user-created database, you must complete both the Database File and Database Name fields. Supply the entire path name.

Drivers for Connections

When you work with a database, all your requests and commands go through a driver to the database itself.

Sybase jConnect JDBC Driver

If you wish to use JDBC from a client application or applet, you must have SAP Sybase jConnect to connect to SAP Sybase IQ databases.

Depending on the installation package you received, SAP Sybase IQ may or may not include SAP Sybase jConnect. You must have jConnect in order to use JDBC from client applications. You can use server-side JDBC without jConnect.

For a full description of jConnect and its use with SAP Sybase IQ, see the jConnect documentation available in the online books or from the *jConnect web site*

Note: Before you can use jConnect in your application, load the driver by entering the statement:

```
Class.forName("com.sybase.jdbc.SybDriver").newInstance();
```

Using the **newInstance** method works around issues in some browsers.

Versions of jConnect Supplied with SAP Sybase IQ

SAP Sybase IQ provides two versions of the Sybase jConnect JDBC driver:

- Full version—If you choose to install jConnect, a jConnect subdirectory is added to your SAP Sybase IQ installation. This holds a directory tree with all jConnect files.
- Zip file—The Remote Data Access features use jConnect to connect to the database. A zip file of the basic jConnect classes is provided to enable jConnect use even without the full development version of the driver.

The jConnect Driver Files

The SAP Sybase jConnect driver is installed into a set of directories under the `jConnect` subdirectory of your SAP Sybase IQ installation. If you have not installed jConnect, you can use the `jdbcdrv.zip` file installed into the `Java` subdirectory.

Classpath Setting for jConnect

For your application to use jConnect, the jConnect classes must be in your CLASSPATH environment variable at compile time and run time, so the Java compiler and Java runtime can locate the necessary files.

For example, the following command adds the jConnect driver class path to an existing CLASSPATH environment variable where *path* is your SAP Sybase IQ installation directory.

```
set classpath=%classpath%;path\jConnect\classes
```

The following alternative command adds the `jdbcdrv.zip` file to your CLASSPATH.

```
set classpath=%classpath%;path\java\jdbcdrv.zip
```

Importing the jConnect Classes

The classes in jConnect are all in the **com.sybase** package. The client application needs to access classes in **com.sybase.jdbc**. For your application to use jConnect, you must import these classes at the beginning of each source file:

```
import com.sybase.jdbc.*
```

Supply URL For the Server

To connect to a database via jConnect, you need to supply a Universal Resource Locator (URL) for the database.

An example given in the section is as follows:

```
StringBuffer temp = new StringBuffer();
// Use the SAP Sybase jConnect driver...
temp.append("jdbc:sybase:Tds:");
// to connect to the supplied machine name...
temp.append(_coninfo);
// on the default port number for ASA...
temp.append(":2638");
// and connect.
System.out.println(temp.toString());
conn = DriverManager.getConnection(temp.toString() , _props );
```

The URL is put together in the following way:

```
jdbc:sybase:Tds:machine-name:port-number
```

The individual components are include:

- jdbc:sybase:Tds—The SAP Sybase jConnect JDBC driver, using the TDS application protocol.
- machine-name—The IP address or name of the machine on which the server is running. If you are establishing a same-machine connection, you can use `localhost`, which means the current machine.
- port number—The port number on which the database server listens. The port number assigned to SAP Sybase IQ is 2638. Use that number unless there are specific reasons not to do so.

The connection string must be less than 253 characters in length.

Specify Database On a Server

Each SAP Sybase IQ server may have one or more databases loaded at a time. The URL as described above specifies a server, but does not specify a database. The connection attempt is made to the default database on the server.

You can specify a particular database by providing an extended form of the URL in one of the following ways.

Using the ServiceName Parameter

```
jdbc:sybase:Tds:machine-name:port-number?ServiceName=DBN
```

The question mark followed by a series of assignments is a standard way of providing arguments to a URL. The case of **ServiceName** is not significant, and there must be no spaces around the = sign. The *DBN* parameter is the database name.

Using the RemotePWD Parameter

A more general method allows you to provide additional connection parameters such as the database name, or a database file, using the **RemotePWD** field. You set **RemotePWD** as a Properties field using the **setRemotePassword()** method.

Here is sample code that illustrates how to use the field.

```
sybDrvr = (SybDriver)Class.forName(
     "com.sybase.jdbc2.jdbc.SybDriver" ).newInstance();
props = new Properties();
props.put( "User", "DBA" );
props.put( "Password", "SQL" );
sybDrvr.setRemotePassword(
    null, "dbf=asiqdemo.db", props );
Connection con = DriverManager.getConnection(
    "jdbc:sybase:Tds:localhost", props );
```

Using the database file parameter **DBF**, you can start a database on a server using jConnect. By default, the database is started with autostop=YES. If you specify a DBF or DBN of **utility_db**, then the utility database is started automatically.

IQ specific connection parameters from TDS clients should be specified in RemotePWD.

This example shows how to specify IQ specific connection parameters, where myconnection becomes the IQ connection name:

```
p.put("RemotePWD","",,CON=myconnection");
```

where myconnection becomes the IQ connection name.

Choose JDBC Driver

The jConnect, SQL Anywhere 16 JDBC 4.0 driver, and the iAnywhere JDBC driver are supported by SAP Sybase IQ. The iAnywhere JDBC driver is deprecated.

Driver	Definition
jConnect	This driver is a 100% pure Java driver. It communicates with SAP Sybase IQ using the TDS client/server protocol.

Driver	Definition
SQL Anywhere 16 JDBC driver	The SQL Anywhere JDBC 4.0 driver is recommended. The SQL Anywhere JDBC 4.0 driver provides some performance benefits and feature benefits compared to the pure Java jConnect JDBC driver; however, this driver does not provide a pure-Java solution.
iAnywhere JDBC driver	This driver is deprecated. The iAnywhere JDBC driver communicates with SAP Sybase IQ using the Command Sequence client/server protocol. Its behavior is consistent with ODBC, embedded SQL, and OLE DB applications. Interactive SQL supports this driver.

For jConnect documentation, see *jConnect for JDBC*.

When choosing which driver to use, you may want to consider the following factors:

- Features—All drivers are JDK 2 compliant. The SQL Anywhere 16 JDBC driver provides fully-scrollable cursors, which are not available in jConnect.
- Pure Java—The jConnect driver is a pure Java solution. The SQL Anywhere 16 JDBC driver requires the SAP Sybase IQ or SQL Anywhere ODBC driver and is not a pure Java solution.
- Performance—The SQL Anywhere 16 JDBC driver provides better performance for most purposes than the jConnect driver.
- Compatibility—The TDS protocol used by the jConnect driver is shared with Adaptive Server Enterprise. Some aspects of the driver's behavior are governed by this protocol, and are configured to be compatible with Adaptive Server Enterprise.

Both drivers are available on Windows 95/98/Me and Windows NT/2000/2003/XP, as well as supported UNIX and Linux operating systems.

JDBC Considerations

Consider the following when running Java applications:

- Java applications running in SAP Sybase IQ run slower than when run outside in a Sun Java Virtual Machine (JVM).

Tip: Tune your applications by increasing the available memory for IQ JVM use with the database options `JAVA_HEAP_SIZE` and `JAVA_NAMESPACE_SIZE`.

Using Server Classes for Remote Data Access

The server class you specify in the **CREATE SERVER** statement determines the behavior of a remote connection. The server classes give the database server detailed server capability information. The database server formats SQL statements specific to a server's capabilities.

There are two categories of server classes:

- ODBC-based server classes
- JDBC-based server classes (deprecated)

Each server class has a set of unique characteristics that configure the server for remote data access.

Refer both to information generic to the server class category (JDBC-based or ODBC-based), and to the information specific to the individual server class.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

ODBC-Based Server Classes

The ODBC-based server classes include:

- saodbc
- ulodbc
- adsodbc
- aseodbc
- db2odbc
- iqodbc
- msaccessodbc
- mssodbc
- mysqlodbc
- odbc
- oraodbc

Note: When using remote data access, if you use an ODBC driver that does not support set conversion is not performed on data coming from that ODBC driver.

Defining ODBC External Servers

The most common way of defining an ODBC-based server is to base it on an ODBC data source. To do this, you can create a data source using the ODBC Administrator.

Once you have defined the data source, the USING clause in the CREATE SERVER statement should match the ODBC data source name.

For example, to configure a IBM DB2 server named mydb2 whose data source name is also mydb2, use:

```
CREATE SERVER mydb2
CLASS 'db2odbc'
USING 'mydb2';
```

Using Connection Strings Instead of Data Sources

An alternative, which avoids using data sources, is to supply a connection string in the **USING** clause of the **CREATE SERVER** statement. To do this, you must know the connection parameters for the ODBC driver you are using. For example, a connection to a remote database may be as follows:

```
CREATE SERVER TestAA
CLASS 'saodbc'
USING 'DRIVER=Sybase IQ 16;HOST=myhost;Server=TestAA;DBN=sample';
```

This defines a connection to a database server named TestAA, running on a computer called myhost, and a database named sample using the TCP/IP protocol.

USING Parameter in the CREATE SERVER Statement

You must run a separate CREATE SERVER statement for each remote database you intend to access.

For example, if a remote server named TestAA is running on the computer banana and owns three databases (db1, db2, db3), you would configure the local database server similar to this:

```
CREATE SERVER TestAAdb1
CLASS 'saodbc'
USING 'banana:2638/db1'
CREATE SERVER TestAAdb2
CLASS 'saodbc'
USING 'banana:2638/db2'
CREATE SERVER TestAAdb3
CLASS 'saodbc'
USING 'banana:2638/db3';
```

If you do not specify a `/database-name` value, the remote connection uses the remote database server's default database.

Server Class saodbc

A server with server class saodbc is a SQL Anywhere database server. No special requirements exist for the configuration of a SQL Anywhere data source.

To access SQL Anywhere database servers that support multiple databases, create an ODBC data source name defining a connection to each database. Issue a **CREATE SERVER** statement for each of these ODBC data source names.

Server Class ulodbc

A server with server class ulodbc is an UltraLite database. Create an ODBC data source name defining a connection to the UltraLite database. Issue a **CREATE SERVER** statement for the ODBC data source name.

There is a one to one mapping between the UltraLite and SAP Sybase IQ data types because UltraLite supports a subset of the data types available in SAP Sybase IQ.

Note: You cannot create a remote server for an UltraLite database running on Mac OS X.

Server Class adsodbc

When you issue a **CREATE TABLE** statement, the database server automatically converts the data types to the corresponding Advantage Database Server data types using the following data type conversions.

SAP Sybase IQ Data Type	ADS Default Data Type
BIT	Logical
TINYINT, SMALLINT, INT, INTEGER	Integer

SAP Sybase IQ Data Type	ADS Default Data Type
BIGINT	Numeric(32)
DECIMAL(p,s), NUMERIC(p,s)	Numeric(p+3)
DATE	Date
TIME	Time
DATETIME, TIMESTAMP	TimeStamp
MONEY, SMALLMONEY	Money
FLOAT, REAL	Double
CHAR(n), VARCHAR(n), LONG VARCHAR	Char(n)
BINARY(n), VARBINARY(n), LONG BINARY	Blob

Server Class aseodbc

A server with server class aseodbc is an Adaptive Server Enterprise (version 10 and later) database server.

The database server requires the installation of the Adaptive Server Enterprise ODBC driver and Open Client connectivity libraries to connect to a remote Adaptive Server Enterprise server with class aseodbc, but the performance is better than with the ASEJDBC class.

Notes:

- Open Client should be version 11.1.1, EBF 7886 or later. Install Open Client and verify connectivity to the Adaptive Server Enterprise server before you install ODBC and configure SAP Sybase IQ. The SAP Sybase ODBC driver should be version 11.1.1, EBF 7911 or later.
- The local setting of the quoted_identifier option controls the use of quoted identifiers for Adaptive Server Enterprise. For example, if you set the quoted_identifier option to Off locally, then quoted identifiers are turned off for Adaptive Server Enterprise.
- Configure a user data source in the Configuration Manager with the following attributes:
  - **General tab –** Type any value for Data Source Name. This value is used in the **USING** clause of the **CREATE SERVER** statement. The server name should match the name of the server in the SAP Sybase interfaces file.
  - **Advanced tab –** Select the **Application Using Threads** and **Enable Quoted Identifiers** options.
  - **Connection tab –** Set the charset field to match your character set. Set the language field to your preferred language for error messages.
  - **Performance tab –** Set the **Prepare Method** to **2-Full**. Set the **Fetch Array Size** as large as possible for the best performance. This increases memory requirements since this is the number of rows that must be cached in memory. Adaptive Server Enterprise recommends using a value of 100. Set **Select Method** to **0-Cursor**. Set **Packet Size** to

as large a value as possible. Adaptive Server Enterprise recommends using a value of -1. Set **Connection Cache** to 1.

Data Type Conversions: ODBC and Adaptive Server Enterprise

When you issue a **CREATE TABLE** statement, the database server automatically converts the data types to the corresponding Adaptive Server Enterprise data types. The following table describes the SAP Sybase IQ to Adaptive Server Enterprise data type conversions.

SAP Sybase IQ Data Type	Adaptive Server Enterprise Default Data Type
BIT	bit
TINYINT	tinyint
SMALLINT	smallint
INT	int
INTEGER	integer
DECIMAL [defaults p=30, s=6]	numeric(30,6)
DECIMAL(128,128)	not supported
NUMERIC [defaults p=30 s=6]	numeric(30,6)
NUMERIC(128,128)	not supported
FLOAT	real
REAL	real
DOUBLE	float
SMALLMONEY	numeric(10,4)
MONEY	numeric(19,4)
DATE	datetime
TIME	datetime
TIMESTAMP	datetime
SMALLDATETIME	datetime
DATETIME	datetime
**BIGDATE	datetime
**BIGDATETIME	datetime

SAP Sybase IQ Data Type	Adaptive Server Enterprise Default Data Type
CHAR(n)	varchar(n)
CHARACTER(n)	varchar(n)
VARCHAR(n)	varchar(n)
CHARACTER VARYING(n)	varchar(n)
LONG VARCHAR	text
TEXT	text
BINARY(n)	binary(n)
LONG BINARY	image
BIGINT	numeric(20,0)

Server Class dbsodbc

A server with server class db2odbc is IBM DB2.

Notes:

- SAP Sybase certifies the use of IBM's DB2 Connect version 5, with fix pack WR09044. Configure and test your ODBC configuration using the instructions for that product. SAP Sybase IQ has no specific requirements for the configuration of IBM DB2 data sources.
- The following is an example of a CREATE EXISTING TABLE statement for a IBM DB2 server with an ODBC data source named mydb2:

```
CREATE EXISTING TABLE ibmcol
AT 'mydb2..sysibm.syscolumns';
```

Data Type Conversions: IBM DB2

When you issue a **CREATE TABLE** statement, the database server automatically converts the data types to the corresponding IBM DB2 data types. The following table describes the SAP Sybase IQ to IBM DB2 data type conversions.

SAP Sybase IQ Data Type	IBM DB2 Default Data Type
BIT	smallint
TINYINT	smallint
SMALLINT	smallint
INT	int
INTEGER	int

SAP Sybase IQ Data Type	IBM DB2 Default Data Type
BIGINT	decimal(20,0)
CHAR(1-254)	varchar(n)
CHAR(255-4000)	varchar(n)
CHAR(4001-32767)	long varchar
CHARACTER(1-254)	varchar(n)
CHARACTER(255-4000)	varchar(n)
CHARACTER(4001-32767)	long varchar
VARCHAR(1-4000)	varchar(n)
VARCHAR(4001-32767)	long varchar
CHARACTER VARYING(1-4000)	varchar(n)
CHARACTER VARYING(4001-32767)	long varchar
LONG VARCHAR	long varchar
TEXT	long varchar
BINARY(1-4000)	varchar for bit data
BINARY(4001-32767)	long varchar for bit data
LONG BINARY	long varchar for bit data
IMAGE	long varchar for bit data
DECIMAL [defaults p=30, s=6]	decimal(30,6)
NUMERIC [defaults p=30 s=6]	decimal(30,6)
DECIMAL(128, 128)	NOT SUPPORTED
NUMERIC(128, 128)	NOT SUPPORTED
REAL	real
FLOAT	float
DOUBLE	float
SMALLMONEY	decimal(10,4)
MONEY	decimal(19,4)
DATE	date

SAP Sybase IQ Data Type	IBM DB2 Default Data Type
TIME	time
SMALLDATETIME	timestamp
DATETIME	timestamp
TIMESTAMP	timestamp

Server Class iqodbc

A server with server class iqodbc is a SAP Sybase IQ server. No special requirements exist for the configuration of a SAP Sybase IQ data source.

To access SAP Sybase IQ servers that support multiple databases, create an ODBC data source name defining a connection to each database. Issue a **CREATE SERVER** statement for each of these ODBC data source names.

**Server class iqodbc uses the same parameters and format as server class saodbc.

Server Class msaccessodbc

Access databases are stored in a `.mdb` file. Using the ODBC manager, create an ODBC data source and map it to one of these files. A new `.mdb` file can be created through the ODBC manager. This database file becomes the default if you don't specify a different default when you create a table through SAP Sybase IQ.

Assuming an ODBC data source named access, you can use any of the following statements to access data:

```
CREATE TABLE tab1 (a int, b char(10))
AT 'access...tab1';
```

```
CREATE TABLE tab1 (a int, b char(10))
AT 'access;d:\pcdb\data.mdb;;tab1';
```

```
CREATE EXISTING TABLE tab1
AT 'access;d:\pcdb\data.mdb;;tab1';
```

Access does not support the owner name qualification; leave it empty.

Data Type Conversions: Microsoft Access

SAP Sybase IQ Data Type	Microsoft Access Default Data Type
BIT, TINYINT	TINYINT
SMALLINT	SMALLINT
INT, INTEGER	INTEGER
BIGINT	DECIMAL(19,0)

SAP Sybase IQ Data Type	Microsoft Access Default Data Type
DECIMAL(p,s), NUMERIC(p,s)	DECIMAL(p,s)
DATE, TIME, DATETIME, TIMESTAMP	DATETIME
MONEY, SMALLMONEY	MONEY
FLOAT	FLOAT
REAL	REAL
CHAR(n), VARCHAR(n)	CHARACTER(n) if n is less than 254 TEXT if n is greater than or equal to 254
LONG VARCHAR	TEXT
BINARY, VARBINARY	BINARY(n) if n is less than 4000 IMAGE if n is greater than or equal to 4000
LONG BINARY	IMAGE

Server Class mssodbc

A server with server class mssodbc is Microsoft SQL Server version 6.5, Service Pack 4.

Notes:

- SAP Sybase certifies the use of Microsoft SQL Server version 3.60.0319 ODBC driver (included in the MDAC 2.0 release). Configure and test your ODBC configuration using the instructions for that product.
- The following is an example of a **CREATE EXISTING TABLE** statement for a Microsoft SQL Server named mymssql:

  ```
  CREATE EXISTING TABLE accounts,
  AT 'mymssql.database.owner.accounts';
  ```
- The local setting of the quoted_identifier option controls the use of quoted identifiers for Microsoft SQL Server. For example, if you set the quoted_identifier option to Off locally, then quoted identifiers are turned off for Microsoft SQL Server.

SAP Sybase IQ Data Type	Microsoft SQL Server Default Data Type
BIT	bit
TINYINT	tinyint
SMALLINT	smallint
INT	int

SAP Sybase IQ Data Type	Microsoft SQL Server Default Data Type
BIGINT	numeric(20,0)
DECIMAL [defaults p=30, s=6]	decimal(prec, scale)
NUMERIC [defaults p=30 s=6]	numeric(prec, scale)
FLOAT	if (prec) float(prec) else float
REAL	real
SMALLMONEY	smallmoney
MONEY	money
DATE	datetime
TIME	datetime
TIMESTAMP	datetime
SMALLDATETIME	datetime
DATETIME	datetime
CHAR(n)	if (length > 255) text else varchar(length)
CHARACTER(n)	char(n)
VARCHAR(n)	if (length > 255) text else varchar(length)
LONG VARCHAR	text
BINARY(n)	if (length > 255) image else binary(length)
LONG BINARY	image
DOUBLE	float
UNIQUEIDENTIFIERSTR	uniqueidentifier

Server Class mysqlodbc

When you issue a **CREATE TABLE** statement, the database server automatically converts the data types to the corresponding MySQL data types using the following data type conversions.

Data Type Conversions: Microsoft SQL Server

SAP Sybase IQ Data Type	MySQL Default Data Type
BIT	bit(1)
TINYINT	tinyint unsigned

SAP Sybase IQ Data Type	MySQL Default Data Type
SMALLINT	smallint
INT, INTEGER	int
BIGINT	bigint
DECIMAL(p,s), NUMERIC(p,s)	decimal(p,s)
DATE	date
TIME	time
DATETIME, TIMESTAMP	datetime
MONEY	decimal(19,4)
SMALLMONEY	decimal(10,4)
FLOAT	float
REAL	real
CHAR(n)	char(n) if n is less than 254 varchar(n) if n is greater than or equal to 254 but less than 4000 longtext if n is greater than or equal to 4000
VARCHAR(n)	varchar(n) if n is less than 4000 longtext if n is greater than or equal to 4000
LONG VARCHAR	longtext
BINARY(n), VARBINARY(n)	varbinary(n) if n is less than 4000 longblob if n is greater than or equal to 4000
LONG BINARY	longblob

Server Class odbc

ODBC data sources that do not have their own server class use server class odbc. You can use any ODBC driver.

SAP Sybase IQ certifies the following ODBC data sources:

- Microsoft Excel (Microsoft 3.51.171300)
- Microsoft FoxPro (Microsoft 3.51.171300)
- Lotus Notes SQL 2.0

The latest versions of Microsoft ODBC drivers can be obtained through the Microsoft Data Access Components (MDAC) distribution found at the Microsoft Download Center. The Microsoft driver versions listed above are part of MDAC 2.0.

Microsoft Excel (Microsoft 3.51.171300)

With Excel, each Excel workbook is logically considered to be a database holding several tables. Tables are mapped to sheets in a workbook. When you configure an ODBC data source name in the ODBC driver manager, you specify a default workbook name associated with that data source. However, when you issue a CREATE TABLE statement, you can override the default and specify a workbook name location string. This allows you to use a single ODBC DSN to access all of your excel workbooks.

In this example, a remote server named excel was created. To create a workbook named work1.xls with a sheet (table) called mywork:

```
CREATE TABLE mywork (a int, b char(20))
AT 'excel;d:\work1.xls;;mywork';
```

To create a second sheet (or table) execute a statement such as:

```
CREATE TABLE mywork2 (x float, y int)
AT 'excel;d:\work1.xls;;mywork2';
```

You can import existing worksheets into the database server using **CREATE EXISTING**, under the assumption that the first row of your spreadsheet contains column names.

```
CREATE EXISTING TABLE mywork
AT'excel;d:\work1;;mywork';
```

If the database server reports that the table is not found, you may need to explicitly state the column and row range you want to map to. For example:

```
CREATE EXISTING TABLE mywork
AT 'excel;d:\work1;;mywork$';
```

Adding the $ to the sheet name indicates that the entire worksheet should be selected.

Note in the location string specified by AT that a semicolon is used instead of a period for field separators. This is because periods occur in the file names. Excel does not support the owner name field so leave this blank.

Deletes are not supported. Also some updates may not be possible since the Excel driver does not support positioned updates.

Microsoft FoxPro (Microsoft 3.51.171300)

You can store FoxPro tables together inside a single FoxPro database file (`.dbc`), or, you can store each table in its own separate `.dbf` file. When using `.dbf` files, be sure the file name is filled into the location string; otherwise the directory that SAP Sybase IQ was started in is used.

```
CREATE TABLE fox1 (a int, b char(20))
AT 'foxpro;d:\pcdb;;fox1';
```

This statement creates a file named `d:\pcdb\fox1.dbf` when you choose the Free Table Directory option in the ODBC Driver Manager.

Lotus Notes SQL 2.0

To obtain this driver, go to the Lotus web site at http://www.lotus.com/. Read the documentation that is included with it for an explanation of how Notes data maps to relational tables. You can easily map SAP Sybase IQ tables to Notes forms.

Here is how to set up the database server to access the Address sample file.

- Create an ODBC data source using the NotesSQL driver. The database will be the sample names file: `c: \notes\data\names.nsf`. The Map Special Characters option should be turned on. For this example, the Data Source Name is `my_notes_dsn`.
- Create a server in SAP Sybase IQ:

```
CREATE SERVER names
CLASS 'odbc'
USING 'my_notes_dsn';
```

- Map the Person form into a SAP Sybase IQ table:

```
CREATE EXISTING TABLE Person
AT 'names...Person';
```

- Query the table:

```
SELECT * FROM Person;
```

Avoiding Password Prompts

Lotus Notes does not support sending a user name and password through the ODBC API. If you try to access Lotus notes using a password protected ID, a window appears on the computer where SAP Sybase IQ is running, and prompts you for a password. Avoid this behavior in multi-user server environments.

To access Lotus Notes unattended, without ever receiving a password prompt, you must use a nonpassword- protected ID. You can remove password protection from your ID by clearing it (choose **File » Tools » User ID » Clear Password**), unless your Domino administrator required a password when your ID was created. In this case, you will not be able to clear it.

Server Class oraodbc

A server with server class oraodbc is Oracle version 8.0 or later.

Notes:

- SAP Sybase certifies the use of the Oracle version 8.0.03 ODBC driver. Configure and test your ODBC configuration using the instructions for that product.
- The following is an example of a **CREATE EXISTING TABLE** statement for an Oracle server named myora:

```
CREATE EXISTING TABLE employees
AT 'myora.database.owner.employees';
```

- As a result of Oracle ODBC driver restrictions, you cannot issue a **CREATE EXISTING TABLE** statement for system tables. A message returns stating that the table or columns cannot be found.

Data Type Conversions: Oracle

When you issue a **CREATE TABLE** statement, the database server automatically converts the data types to the corresponding Oracle data types using the following data type conversions.

SAP Sybase IQ Data Type	Oracle Data Type
BIT	number(1,0)
TINYINT	number(3,0)
SMALLINT	number(5,0)
INT	number(11,0)
BIGINT	number(20,0)
DECIMAL(prec, scale)	number(prec, scale)
NUMERIC(prec, scale)	number(prec, scale)
FLOAT	float
REAL	real
SMALLMONEY	numeric(13,4)
MONEY	number(19,4)
DATE	date
TIME	date
TIMESTAMP	date
SMALLDATETIME	date
DATETIME	date
CHAR(n)	if (n > 255) long else varchar(n)
VARCHAR(n)	if (n > 2000) long else varchar(n)
LONG VARCHAR	long or clob
BINARY(n)	if (n > 255) long raw else raw(n)
VARBINARY(n)	if (n > 255) long raw else raw(n)
LONG BINARY	long raw

JDBC-Based Server Classes (Deprecated)

Support for the JDBC-based server classes is deprecated. You should update your applications to use the ODBC-based server classes.

JDBC-based server classes are used when SAP Sybase IQ internally uses a Java VM and jConnect 5.5 to connect to the remote server. The JDBC-based server classes are:

- **SAJDBC –** SQL Anywhere.
- **ASEJDBC –** Adaptive Server Enterprise (version 10 and later).
- **IQJDBC –** SAP Sybase IQ.

Configuration Notes for JDBC Classes

When you access remote servers defined with JDBC-based classes, consider that:

- For optimum performance, an ODBC-based class is recommended (saodbc or aseodbc).
- Any remote server that you access using the ASEJDBC or SAJDBC server class must be set up to handle a jConnect 6.x-based client.
- If a JDBC remote server connection is disconnected or lost, you only find out that the server is unavailable if you attempt to use the JDBC remote server to access a proxy object, such as a proxy table or proxy procedure. ODBC does not have this limitation.

Server Class SAJDBC (Deprecated)

This server class is deprecated. You should update your application to use server class saodbc.

A server with server class SAJDBC is a SQL Anywhere server. No special requirements exist for the configuration of a SQL Anywhere data source.

Server Class ASEJDBC (deprecated)

This server class is deprecated. You should update your application to use server class aseodbc.

A server with server class ASEJDBC is an Adaptive Server Enterprise (version 10 and later) server. No special requirements exist for the configuration of an Adaptive Server Enterprise data source.

Note: The local setting of the quoted_identifier option controls the use of quoted identifiers for Adaptive Server Enterprise. For example, if you set the quoted_identifier option to Off locally, then quoted identifiers are turned off for Adaptive Server Enterprise.

Data Type Conversions: JDBC and Adaptive Server Enterprise

When you issue a **CREATE TABLE** statement, SQL Anywhere automatically converts the data types to the corresponding Adaptive Server Enterprise data types using the following data type conversions.

SAP Sybase IQ Data Type	Adaptive Server Enterprise Default Data Type
BIT	bit
TINYINT	tinyint
SMALLINT	smallint
INT	int
INTEGER	integer
DECIMAL [defaults p=30, s=6]	numeric(30,6)
DECIMAL(128,128)	not supported
NUMERIC [defaults p=30 s=6]	numeric(30,6)
NUMERIC(128,128)	not supported
FLOAT	real
REAL	real
DOUBLE	float
SMALLMONEY	numeric(10,4)
MONEY	numeric(19,4)
DATE	datetime
TIME	datetime
TIMESTAMP	datetime
SMALLDATETIME	datetime
DATETIME	datetime
CHAR(n)	varchar(n)
CHARACTER(n)	varchar(n)
VARCHAR(n)	varchar(n)
CHARACTER VARYING(n)	varchar(n)
LONG VARCHAR	text
TEXT	text
BINARY(n)	binary(n)

SAP Sybase IQ Data Type	Adaptive Server Enterprise Default Data Type
LONG BINARY	image
IMAGE	image
BIGINT	numeric(19,0)

Server Class IQJDBC (deprecated)

This server class is deprecated. You should update your application to use server class iqodbc.

A server with server class IQJDBC is a SAP Sybase IQ server. No special requirements exist for the configuration of a SAP Sybase IQ data source.

ODBC Data Sources

You can store a set of SAP Sybase IQ connection parameters as a data source in either the system registry or as a file. Data sources are required to use applications that connect using the Open Database Connectivity (ODBC) interface.

Microsoft Corporation defines the ODBC interface, which is a standard interface for connecting client applications to database management systems in the Windows environments. Many client applications, including application development systems, use the ODBC interface to access a wide range of database systems.

On Windows, the ODBC Administrator provides a central place for creating and managing ODBC data sources.

Note: Use the 32-bit ODBC Administrator to manage data sources when using the 32-bit ODBC drivers and the 64-bit ODBC Administrator with 64-bit clients.

The following procedure uses the ODBC Administrator to add a new data source to your existing `odbc.ini`, or creates a new file if necessary.

SAP Sybase IQ also includes a cross-platform command-line utility named `iqdsn` to create data sources.

Before You Begin

This section describes how to create an ODBC data source. Before you create a data source, you need to know which connection parameters you want to include in it.

In ODBC Administrator, you can work with User Data Sources, File Data Sources, and System Data Sources.

See also

- *Roadmap for Connections* on page 41

- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Where Data Sources Are Held

Data sources can be used on Windows or UNIX/Linux-based systems to help clients make it easier to make connections.

When you connect to a database using ODBC, you use an ODBC data source. The data source contains a set of connection parameters. You need an ODBC data source on the client computer for each database to which you connect.

If you have a data source, your connection string can simply name the data source to use:

- Data source – Use the DSN connection parameter to reference a user or system data source:

  ```
  DSN=my data source
  ```

  On Windows, user and system data sources are stored in the registry and in the file `odbc.ini`. On UNIX platforms user data sources are in the file `.odbc.ini`.
- File data source – Use the FileDSN connection parameter to reference a data source held in a file:

  ```
  FileDSN=mysource.dsn
  ```

Except for encrypted passwords, which are allowed in FileDSNs only, you can put identical connection information in DSNs and FileDSNs.

ODBC data sources can be used to help a wide range of clients connect including:

- Applications on Windows, Linux and Unix that use the ODBC interface
- Java applications using the iAnywhere JDBC driver

Creating a Data Source from the ODBC Administrator

The ODBC Administrator provides an easy way to create an ODBC Data Source.

1. Start the ODBC Administrator:

   Select **Settings > Control Panel > Administrative Tools > Data Sources (ODBC)**.

   or

   Sybase IQ 16.0 > ODBC Administrator 32-bit

   The ODBC Data Source Administrator appears.
2. Click **Add**.

   The Create New Data Source wizard appears.
3. Select the SAP Sybase IQ from the list of drivers and click **Finish**.
4. In the ODBC Configuration window, type the Data Source Name.
5. Now click the Login tab. Type the User ID and Password for your database. For example, use `DBA` and `sql`.
6. Click the Database tab. If the data source is on your local machine, type the Server name, a Start line and Database file, including the path.
7. If the data source is on a remote system, click the Network tab. Click the box for the appropriate protocol and specify the options beside the box. For example, to connect to a server using TCP/IP protocol and port 1870, you would click **TCP/IP** and type:

   ```
   port=1870
   ```
8. Click **OK** when you have finished defining your data source.

   The ODBC Data Source Administrator returns you to the User DSN tab.

Note: When specifying network connections, you need a different *systemname*:*port#* combination for each database server. The port number must match the one you started the server with.

Creating an ODBC Data Source from the Command Line

You can create User and System Data Sources using the `iqdsn` command-line utility. You cannot create File Data Sources with `iqdsn`. You can also use the ODBC Administrator to create User, System, and File Data Sources.

1. Open a command prompt.
2. Enter an **iqdsn** command specifying connection parameters.

   For example, the following command creates a data source for the SAP Sybase IQ demo database. The command must be entered on one line:

```
iqdsn -w "My DSN"
"uid=DBA;pwd=sql;dbf=c:\Program Files\Sybase\IQ-16_0\demo
\iqdemo.db"
```

The **iqdsn** output contains the following line:

```
User Data Source "My DSN" written to registry.
```

The **iqdsn** utility lists SAP Sybase IQ User Data Sources created on the Windows command line.

For more information on the **iqdsn** utility, see *Utility Guide > iqdsn Database Administration Utility*.

To edit a data source, select one from the list in the ODBC administrator window and click **Configure**.

To access Windows across a network to create an ODBC data source, see the *Installation and Configuration Guide*.

Testing an ODBC Data Source

You can test ODBC Data Sources using the ODBC Administrator.

1. Start the database. For example, to start the demo database, select Programs > Sybase > Sybase IQ 16.0 > Start Sybase IQ Demo Database.
2. In the ODBC Data Source Administrator, select your new data source from the list of User Data Sources.
3. Click Configure.
4. On the ODBC Configuration dialog box, click Test Connection.

   If you cannot access the Data Source, check that you have filled out the various tabs with correct file and path names.

Configuring ODBC Data Sources in ODBC Administrator

Options on the ODBC configuration dialog are organized by tab.

Except for the ODBC tab, options on the tabs correspond to SAP Sybase IQ connection parameters.

ODBC Tab

Use the ODBC tab on the ODBC configuration dialog to specify attributes such as the data source name.

Area	Description
ODBC tab – Data source name	The Data Source Name is used to identify the ODBC data source. You can use any descriptive name for the data source (spaces are allowed) but it is recommended that you keep the name short, as you may need to enter it in connection strings.
ODBC tab – Description	You can enter an optional longer description of the data source.
ODBC tab – Isolation level	The isolation level for an SAP Sybase IQ data source is always effectively 3. However, the default catalog store isolation level is 0. You should generally leave this blank.
ODBC tab – Microsoft applications (Keys in SQL Statistics)	Check this box if you wish foreign keys to be returned by SQL statistics. The ODBC specifications states that primary and foreign keys should not be returned by SQL statistics, however, some Microsoft applications (such as Visual Basic and Access) assume that primary and foreign keys are returned by SQL Statistics.
ODBC tab – Delphi applications	Check this box to improve performance for Borland Delphi applications. Checking this option assigns one bookmark value to each row, instead of the two that are otherwise assigned (one for fetching forwards and a different one for fetching backwards).
ODBC tab – Suppress fetch warnings	Check this box to suppress warning messages that are returned from the database server on a fetch.
ODBC tab – Prevent driver not capable errors	The SAP Sybase IQ 16 ODBC driver returns a `Driver not capable` error code because it does not support qualifiers. Some ODBC applications do not handle this error properly. Check this box to disable this error code, allowing such applications to work.
ODBC tab – Delay AutoCommit until statement close	Force the SAP Sybase IQ 16 ODBC driver to delay the commit operation until a statement has been closed.
ODBC tab – Describe cursor behavior	Select how often you wish a cursor to be re-described when a procedure is executed or resumed.
ODBC tab – Translator	Choose MS Code Page Translator if your database uses an OEM code page. If your database uses an ANSI code page, which is the default, leave this unchecked.
ODBC tab – Test Connection	Test that the information provided results in a proper connection. In order for the test to work, you must have specified a user ID and password.

ODBC Data Sources on UNIX

On UNIX-like operating systems, the `odbc.ini` file contains a list of data sources.

When you create an `.odbc.ini` file, use the long form of each identifier, for example:

```
[My Data Source]
EngineName=myserver
CommLinks=tcpip(port=1870)
Userid=DBA
Password=sql
```

The preceding sample DSN shows only a subset of the valid connection parameters. Add network communication parameters as part of the CommLinks (LINKS) parameter.

You can create and manage ODBC data sources on UNIX using the **iqdsn** command-line utility.

ODBC Data Source File Location

References to ODBC functions are resolved at runtime.

To connect with ODBC data sources, the location of your `.odbc.ini` file must be referenced by one of the following variables. SAP Sybase IQ searches the directories specified by the variables below in the following order:

1. $ODBCINI – must contain the exact full path name of the `.odbc.ini` file.
2. $HOME
3. Current directory
4. $PATH

SAP Sybase IQ clients ignore the following variables when searching for `.odbc.ini`:

1. $ODBC_HOME
2. $ODBC_INI

Use a text editor to edit `.odbc.ini`.

On UNIX-like operating systems, SAP Sybase IQ installs an ODBC driver and driver manager. The name of the driver file includes an operating system-specific extension, for example, `so` for Solaris systems. If you are using an ODBC application that uses `libodbc.so` (`libodbc.so.1`) or `libodbcinst.so` (`libodbcinst.so.1`), simply create symbolic links for these that point to `$SYBASE/IQ-16_0/lib64/libdbodbc16.so.1`. If you are creating a custom ODBC application, you can link directly to `libdbodbc16.so`.

If an ODBC driver manager is not present, the IQ ODBC driver (found via the symbolic link) uses the `.odbc.ini` for data source information.

ODBC Trace Output

To create an ODBC trace file, use the unixODBC driver manager.

Use the `libdbodbc16.so` driver and leave it up to the driver to choose the multithreaded or unthreaded driver. Tracing capability exists in the switch (`libdbodbc16.so`), not in the individual drivers (`libdbodbc16_n.so` or `libdodbc16_r.so`). If you change the driver to point to the `_r` version, you remove the switch from the call sequence, preventing the tracing.

The unixODBC driver manager

Versions of the unixODBC release before version 2.2.14 have incorrectly implemented some aspects of the 64-bit ODBC specification as defined by Microsoft. These differences will cause problems when using the unixODBC driver manager with the SAP Sybase IQ 64-bit ODBC driver.

To avoid these problems, you should be aware of the differences. One of them is the definition of SQLLEN and SQLULEN. These are 64-bit types in the Microsoft 64-bit ODBC specification, and are expected to be 64-bit quantities by the SAP Sybase IQ 64-bit ODBC driver. Some implementations of unixODBC define these two types as 32-bit quantities and this will result in problems when interfacing to the SAP Sybase IQ 64-bit ODBC driver.

There are three things that you must do to avoid problems on 64-bit platforms.

1. Instead of including the unixODBC headers like `sql.h` and `sqlext.h`, you should include the SAP Sybase IQ ODBC header file `unixodbc.h`. This will guarantee that you have the correct definitions for SQLLEN and SQLULEN. The header files in unixODBC 2.2.14 or later versions correct this problem.
2. You must ensure that you have used the correct types for all parameters. Use of the correct header file and the strong type checking of your C/C++ compiler should help in this area. You must also ensure that you have used the correct types for all variables that are set by the SAP Sybase IQ driver indirectly through pointers.
3. Do not use versions of the unixODBC driver manager before release 2.2.14. Link directly to the SAP Sybase IQ ODBC driver instead. For example, ensure that the libodbc shared object is linked to the SAP Sybase IQ driver.

   ```
   libodbc.so.1 -> libdbodbc16_r.so.1
   ```

   Alternatively, you can use the SAP Sybase IQ driver manager on platforms where it is available.

File Data Sources

On Windows operating systems, ODBC data sources are typically stored in the system registry. File data sources are an alternative, which are stored as files. File data sources are supported on both Windows and UNIX systems.

In Windows, file data sources typically have the extension `.dsn`. They consist of sections, each section starting with a name enclosed in square brackets. DSN files are very similar in layout to initialization files.

To connect using a File Data Source, use the **FileDSN** connection parameter. You cannot use both **DSN** and **FileDSN** in the same connection.

File Data Sources Can Be Distributed

One benefit of file data sources is that you can distribute the file to users, so that connection information does not have to be reconstructed on each machine. If the file is placed in the default location for file data sources, it is picked up automatically by ODBC. In this way, managing connections for many users can be made simpler.

Note: Because DSNs are stored in the Windows registry, they are public information. For this reason you should not put a password in a DSN, unless you encrypt it. If you want to store your password in your data source, use a File DSN.

Embedded SQL applications can also use ODBC file data sources.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Creating a File Data Source Using the ODBC Administrator

You can use the ODBC Administrator tool to create a File Data Source.

1. Start the ODBC Administrator, click the File DSN tab and click Add.
2. Select SAP Sybase IQ from the list of drivers, and click Next.

3. Follow the instructions to create the data source.

File Data Sources and Text Editors

A file data source is a text file that can be edited with any text editor. One limitation to using a text editor is that you cannot store encrypted passwords in the file.

Example of a File Data Source

```
[Sample File Data Source]
 ENG = iqdemo
 DBA = DBA
 PWD = sql
```

See *Utility Guide > iqdsn Database Administration Utility*.

The iAnywhere Solutions 16 Oracle ODBC Driver

The iAnywhere Solutions 16 Oracle ODBC driver is custom-tailored for use with SAP Sybase IQ or SQL Anywhere. It does not work with third-party software.

If you use Oracle with OMNI, you must install an Oracle client on the same computer as this Oracle driver.

Use the ODBC Administrator, the `.odbc.ini` file (in UNIX), or the **iqdsn** utility to configure the Oracle driver.

Table 2. iAnywhere Solutions 16 Oracle ODBC driver configuration options

Windows ODBC Administrator	Configuration for iqdsn command or .odbc.ini file	Description
Data Source Name	For **iqdsn**, use the **-w** option.	A name to identify your data source.
User ID	In **iqdsn**, set the UserID option in the connection string.	The default login ID that the application uses to connect to your Oracle database. If you leave this field blank, you must supply the information when you connect.
Password	In **iqdsn**, set this option in the connection string.	The password that the application uses to connect to your Oracle database. If you leave this field blank, you must supply the information when you connect.

Windows ODBC Administrator	Configuration for iqdsn command or .odbc.ini file	Description
TNS service name	**ServiceName**	The TNS Service Name that is stored in `network/admin/tnsnames.ora` under your Oracle installation directory.
Encrypt Password	For **iqdsn**, use the **-pe** option.Not supported for `.odbc.ini`.	Select this checkbox if you want the password to be stored in encrypted form in the data source.
Procedure returns results or uses VARRAY parameters	• **ProcResults –** In **iqdsn**, set the ProcResults option in the connection string. • **ProcOwner –** In **iqdsn**, set the ProcOwner option in the connection string.	Select this field if your stored procedures can return results or if the stored procedures use Oracle VARRAYs. The default is that this option is not selected. If your download_cursor or download_delete_cursor scripts are stored procedure invocations, select this checkbox. If no stored procedures use VARRAYs and none of them returns a result set, clear this checkbox to improve performance.
Array Size	In **iqdsn**, set the ArraySize option in the connection string.	The size, in bytes, of the byte array used to prefetch rows, on a per-statement basis. The default is 60000. Increasing this value can significantly improve fetch performance at the cost of extra memory allocation.
Enable Microsoft Distributed Transactions	For **iqdsn**, use the **enableMSDIC** option in the connection string. Not supported for `.odbc.ini`.	Select this option to enlist your transactions in the Microsoft Distributed Transaction Coordinator. When selected, the Oracle ODBC driver requires an Oracle binary file, `oramts.dll` for Oracle 9i clients or `oramts10.dll` for Oracle 10g clients.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52

- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Creating an Oracle DSN on UNIX

On UNIX, you may set up the driver in an ODBC system information file (typically called `.odbc.ini`).

1. Use a text editor to edit the system information file. The section for this driver should appear as follows (with appropriate values entered for each field):

```
[sample_dsn_using_the_sa_odbc_driver_for_oracle]
Driver=full-path/libdboraodbc16_r.so
UserID=user-id
Password=password
SID=TNS-service-name
ProcResults=[yes|no]
ArraySize=bytes
```

2. Click **Add**.
3. Choose **iAnywhere Solutions 16 - Oracle**, then click **Finish**.
4. Specify the configuration options.
5. Click **Test Connection**, then click **OK**.

Creating an Oracle DSN on Windows

Use the ODBC Administrator to create an Oracle Data Source Name on Windows.

1. Select **Start > Programs > Sybase > Sybase IQ 16 > ODBC Administrator 32-bit** or **ODBC Administrator 64-bit**.
2. Click **Add**.

3. Choose **iAnywhere Solutions 16 - Oracle**, then click **Finish**.
4. Specify the configuration options.
5. Click **Test Connection**, then click **OK**.

Creating an Oracle DSN Using IQDSN

On UNIX, you may also use the IQDSN utility to create Oracle DSNs.
To create an Oracle DSN with the **iqdsn** utility, use:

```
iqdsn -w data-source-name -or -c configuration-options
```

For example:

```
iqdsn -w MyOracleDSN -or -pe -c
Userid=dba;Password=sql;SID=abcd;ArraySize=100000;ProcResults=y;ena
bleMSDIC=n
```

For example:

```
iqdsn -w MyOracleDSN -or -pe -c
Userid=dba;Password=sql;SID=abcd;ArraySize=100000;ProcResults=y;ena
bleMSDIC=n
```

Database Connections Using OLE DB

OLE DB uses the Component Object Model (COM) to make data from a variety of sources available to applications.

Relational databases are among the classes of data sources that you can access through OLE DB.

This section describes how to connect to an SAP Sybase IQ database using OLE DB from the following environments:

- Sybase PowerBuilder® can access OLE DB data sources, and you can use SAP Sybase IQ as a PowerBuilder OLE DB database profile.
- Microsoft ActiveX Data Objects (ADO) provides a programming interface for OLE DB data sources. You can access RLV from programming tools such as Microsoft Visual Basic.

OLE DB requires a Windows client. However, you can access both Windows and UNIX servers using OLE DB.

This section is an introduction to how to use OLE DB from Sybase PowerBuilder and Microsoft ADO environments such as Visual Basic. It is not complete documentation on how to program using ADO or OLE DB. The primary source of information on development topics is your development tool documentation.

See also

- *Roadmap for Connections* on page 41

- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

OLE DB Providers

You need an *OLE DB provider* for each type of data source you wish to access.

Each provider is a dynamic-link library. There are two OLE DB providers you can use to access SAP Sybase IQ:

- SQL Anywhere OLE DB provider The SQL Anywhere OLE DB provider provides access to SAP Sybase IQ as an OLE DB data source without the need for ODBC components. The short name for this provider is **SAOLEDB**.

  When the **SAOLEDB** provider is installed, it registers itself. This registration process includes making registry entries in the COM section of the registry, so that ADO can locate the DLL when the **SAOLEDB** provider is called. If you change the location of your DLL, you must re-register it.

  If you use the SQL Anywhere OLE DB provider, ODBC is not required in your deployment.
- Microsoft OLE DB provider for ODBC Microsoft provides an OLE DB provider with a short name of **MSDASQL**.

  The **MSDASQL** provider makes ODBC data sources appear as OLE DB data sources. It requires the Sybase IQ ODBC driver.

Connections from ADO

ADO is an object-oriented programming interface. In ADO, the **Connection** object represents a unique session with a data source.

You can use the following **Connection** object features to initiate a connection:

- The **Provider** property that holds the name of the provider. If you do not supply a Provider name, ADO uses the MSDASQL provider.
- The **ConnectionString** property that holds a connection string. This property holds a RLV connection string, which is used in just the same way as the ODBC driver. You can supply ODBC data source names, or explicit UserID, Password, DatabaseName, and other parameters, just as in other connection strings.
- The **Open** method initiates a connection.

Sample code for connecting to a database:

```
Private Sub cmdTestConnection_Click( _
        ByVal eventSender As System.Object, _
        ByVal eventArgs As System.EventArgs) _
        Handles cmdTestConnection.Click

    ' Declare variables
    Dim myConn As New ADODB.Connection
    Dim myCommand As New ADODB.Command
    Dim cAffected As Integer

    On Error GoTo HandleError

    ' Establish the connection
    myConn.Provider = "IQOLEDB"
    myConn.ConnectionString = _
        "Data Source=Sybase IQ Demo"
    myConn.Open()
    MsgBox("Connection succeeded")
    myConn.Close()
    Exit Sub

HandleError:
    MsgBox(ErrorToString(Err.Number))
    Exit Sub
End Sub
```

Connections from Other Databases

You can access data in SAP Sybase IQ tables as a foreign data source from Adaptive Server Enterprise. To take advantage of this feature, use the Component Integration Services (CIS) interface, which makes data from distributed, heterogeneous sources available to clients.

With CIS, define "proxy tables" in Adaptive Server Enterprise that represent your SAP Sybase IQ tables. You can then query the proxy tables from Adaptive Server Enterprise. See *Component Integration Services Users Guide for Adaptive Server Enterprise*.

CIS and SAP Sybase IQ offer several other ways to connect to other databases and share data, so that user applications can access your entire data warehouse through a common interface. Using CIS, you can:

- Access data in an Adaptive Server Enterprise database from SAP Sybase IQ. This functionality is supported on only some platforms. See the *Installation and Configuration Guide* for your platform.
- Access data in SAP Sybase IQ and SQL Anywhere databases on other database servers.
- Access other foreign data sources, including other vendors' relational databases, Excel spreadsheet data, and text files.
- Join tables from separate SAP Sybase IQ databases.

See *Programming > Accessing Remote Data*.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103

- *How to End Connections* on page 105
- *Connection Logging* on page 106

Avoiding Port Number Conflicts on UNIX

Update configuration files to avoid port number conflicts.

1. Add the following line to `$IQDIR16/scripts/default.cfg` with an unused port number, for example:

   ```
   -x  tcpip{port=4444}
   ```
2. Look for a port number definition in each configuration file. For example, `/usr/summers/mydemo/iqdemo.cfg` contains the following line:

   ```
   -x  tcpip{port=2638}
   ```
3. Edit the line and replace the default port number with the new one, for example:

   ```
   -x  tcpip{port=4444}
   ```
4. Save each file when finished.

If SQL Anywhere is on the same subnet as SAP Sybase IQ, the server names must be unique.

How to Test Connections

The **dbping** command-line utility is provided to help troubleshoot connections. In particular, you can use it to test if a server with a certain name is available on your network.

The **dbping** utility takes a connection string as a command-line option, but by default only those pieces required to locate a server are used. It does not attempt to start a server.

The following line tests to see if a server named Ciaran is available over a TCP/IP connection:

```
dbping -c "eng=Ciaran;CommLinks=tcpip"
```

The following command tests to see if a default server is available on the current machine:

```
dbping
```

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79

- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Integrated Logins

The integrated login feature allows you to maintain a single user ID and password for both database connections and operating system and/or network logins.

Operating Systems Supported

Integrated login capabilities are available for the Windows server only. It is possible for clients on supported Windows platforms to use integrated logins to connect to a network server running on Windows.

Benefits of Integrated Logins

An integrated login is a mapping from one or more Windows user profiles to an existing user in a database. A user who has successfully navigated the security for that user profile and logged in to their machine can connect to a database without providing an additional user ID or password.

To accomplish this, the database must be enabled to use integrated logins and a mapping must have been granted between the user profile used to log in to the machine and/or network, and a database user.

Using an integrated login is more convenient for the user and permits a single security system for database and network security. Its advantages include:

- When connecting to a database using an integrated login, the user does not need to enter a user ID or password.
- If you use an integrated login, the user authentication is done by the operating system, not the database: a single system is used for database security and machine or network security.
- Multiple user profiles can be mapped to a single database user ID.
- The name and password used to log in to the Windows machine do not have to match the database user ID and password.

Warning! Integrated logins offer the convenience of a single security system but there are important security implications which database administrators should be familiar with.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Using Integrated Logins

You must perform several steps in order to connect via an integrated login.

1. Enable the integrated login feature in a database by setting the value of the LOGIN_MODE database option to either **Mixed** (deprecated) or **Integrated** (the option is case insensitive), in place of the default value of **Standard**.)This step requires SET ANY SECURITY OPTION system privilege.
2. Create an integrated login mapping between a user profile and an existing database user. This can be done using a SQL statement.
3. Connect from a client application in such a way that the integrated login facility is triggered.

Each of these steps is described in the sections below.

Enabling the Integrated Login Feature

The LOGIN_MODE database option determines whether the integrated login feature is enabled. As database options apply only to the database in which they are found, different

databases can have a different integrated login setting even if they are loaded and running within the same server.

The LOGIN_MODE database option accepts one of following three values (which are case insensitive):

- 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.

Warning! Setting the LOGIN_MODE database option to Integrated restricts connections to only those users who have been granted an integrated login mapping. Attempting to connect using a user ID and password generates an error. The only exception to this are users with MANAGE ANY USER system privilege.

Creating an Integrated Login in SQL

User profiles can only be mapped to an existing database user ID. When that database user ID is removed from the database, all integrated login mappings based on that database user ID are automatically removed.
The following SQL statement allows Windows users fran_whitney and matthew_cobb to log in to the database as the user `DBA`, without having to know or provide the DBA user ID or password.

```
GRANT INTEGRATED LOGIN
TO fran_whitney, matthew_cobb
AS USER DBA
```

For more information, see *Reference: Statements and Options > SQL Statements > GRANT Statement*.

See also

- *GRANT INTEGRATED LOGIN Statement* on page 488

Revoking an Integrated Login Permission (SQL)

You can remove an integrated login mapping using Interactive SQL.

1. Connect to a database as a user with administrative authority over the MANAGE ANY USER system privilege.
2. Execute a REVOKE INTEGRATED LOGIN FROM statement.

Connecting from a Client Application

A client application can connect to a database using an integrated login in one of the following ways:

- Set the INTEGRATED parameter in the list of connection parameters to **yes**.

- Specify neither a user ID nor a password in the connection string or connection dialog. This method is available only for Embedded SQL applications, including the SAP Sybase IQ administration utilities.

If INTEGRATED=**yes** is specified in the connection string, an integrated login is attempted. If the connection attempt fails and the LOGIN_MODE database option is set to **Mixed** (deprecated), the server attempts a standard login.

If an attempt to connect to a database is made without providing a user ID or password, an integrated login is attempted. The attempt succeeds or fails depending on whether the current user profile name matches a integrated login mapping in the database.

Security Concerns: Unrestricted Database Access

The integrated login features works by using the login control system of Windows in place of the system that SAP Sybase IQ uses to control access to the database.

Essentially, you pass through the database security if you can log in to the machine hosting the database, and if other conditions outlined in this chapter are met.

If you successfully log in to the Windows server as “dsmith”, you can connect to the database without any further proof of identification provided there is either an integrated login mapping or a default integrated login user ID.

When using integrated logins, database administrators should give special consideration to the way Windows enforces login security in order to prevent unwanted access to the database.

In particular, be aware that by default a “Guest” user profile is created and enabled when Windows Workstation or Server is installed.

Warning! Leaving the user profile Guest enabled can permit unrestricted access to a database being hosted by that server.

If the Guest user profile is enabled and has a blank password, any attempt to log in to the server will be successful. It is not required that a user profile exist on the server, or that the login ID provided have domain login permissions. Literally any user can log in to the server using any login ID and any password: they are logged in by default to the Guest user profile.

This has important implications for connecting to a database with the integrated login feature enabled.

Consider the following scenario, which assumes the Windows server hosting a database has a “Guest” user profile that is enabled with a blank password.

- An integrated login mapping exists between the user `dsmith` and the database user ID `DBA`. When the user `dsmith` connects to the server with her correct login ID and password, she connects to the database as `DBA`, a user with full administrative rights.
- But anyone else attempting to connect to the server as “dsmith” will successfully log in to the server regardless of the password they provide because Windows will default that connection attempt to the “Guest” user profile. Having successfully logged in to the server

using the "dsmith" login ID, the unauthorized user successfully connects to the database as `DBA` using the integrated login mapping.

Note: Disable the "Guest" user profile for security. The safest integrated login policy is to disable "Guest" on any Windows machine hosting a SAP Sybase IQ database. This can be done using the Windows User Manager utility.

Temporary Public Options Provide Added Security

Setting the value of the LOGIN_MODE option for a given database to **Mixed** (deprecated) or **Integrated** using the following SQL statement permanently enables integrated logins for that database.

```
SET OPTION Public.LOGIN_MODE = Mixed
```

If the database is shut down and restarted, the option value remains the same and integrated logins are still enabled.

Changing the LOGIN_MODE option temporarily will still allow user access via integrated logins. The following statement will change the option value temporarily:

```
SET TEMPORARY OPTION "Public".LOGIN_MODE = Mixed
```

If the permanent option value is **Standard**, the database will revert to that value when it is shut down.

Setting temporary public options can be considered an additional security measure for database access since enabling integrated logins means that the database is relying on the security of the operating system on which it is running. If the database is shut down and copied to another machine (such as a user's machine) access to the database reverts to the SAP Sybase IQ security model and not the security model of the operating system of the machine where the database has been copied.

See *Reference: Statements and Options > SQL Statements > SET OPTION Statement*.

See also

- *SET OPTION Statement* on page 496

Network Aspects of Integrated Logins

If the database is located on a network server, then one of two conditions must be met for integrated logins to be used:

- The user profile used for the integrated login connection attempt must exist on both the local machine and the server. As well as having identical user profile names on both machines, the passwords for both user profiles must also be identical.

  For example, when the user **jsmith** attempts to connect using an integrated login to a database loaded on network server, identical user profile names and passwords must exist

on both the local machine and application server hosting the database. **jsmith** must be permitted to log in to both the local machine and the server hosting the network server.

- If network access is controlled by a Microsoft Domain, the user attempting an integrated login must have domain permissions with the Domain Controller server and be logged in to the network. A user profile on the network server matching the user profile on the local machine is not required.

Default Integrated Login Users

You can create a default integrated login user ID that allows connection via an integrated login to succeed even if no integrated login mapping exists for the user profile currently in use.

For example, if no integrated login mapping exists for the user profile name **jsmith**, an integrated login connection attempt will normally fail when **jsmith** is the user profile in use.

However, if you create a user ID named **Guest** in a database, an integrated login successfully maps to the **Guest** user ID if no integrated login mapping explicitly identifies the user profile **jsmith**.

The default integrated login user permits anyone attempting an integrated login to successfully connect to a database if the database contains a user ID named **Guest**. The permissions and privileges granted to the newly-connected user are determined by the privileges granted to the **Guest** user ID.

Connection Pooling

Connection pooling may improve the performance of applications that make multiple, brief connections to the database server. If connection pooling is enabled for a connection, when it is disconnected, the connection is automatically cached and may be reused when the application reconnects. You control connection pooling with the ConnectionPool (CPOOL) connection parameter. Once an application makes a specified number of connections with the same connection string, then the connection is pooled

An application must make five connections with the same connection string before a connection is cached. The connection name can be different each time, but all other connection parameters must be identical for a cached connection to be reused.

If the application process connects again and there are cached connections available for the same connection string, the cached connection is reused. Connections remain in the cached state for the time specified by the ConnectionPool (CPOOL) connection parameter (60 seconds by default).

Cached connections are not reused if it would change the behavior of the application. For example, cached connections are not reused:

- For databases that stop automatically when there are no connections to them.
- If connections are disabled.

- If the database server has reached its connection limit.
- If a password has changed.
- If the login_mode option is set.

To ensure that connection pooling is transparent to the application, a connection is disconnected if a failure occurs when caching a connection. If a failure occurs when attempting to reuse a cached connection, the database server attempts to connect normally

A connection is cached if it is disconnected and the maximum number of connections specified by the CPOOL connection parameter has not been reached. The connection is reinitialized, and the cached connection remains connected to the database server even though the application has disconnected it. The cleanup and reinitialization of a connection includes the following activities:

- Rolling back all outstanding transactions.
- Dropping temporary tables, temporary functions, and variables.
- Resetting connection options and connection counters.
- Decrementing and incrementing the database server connection counts. You are not informed that there are active connections when a database server with only cached connections shuts down.
- Executing all defined disconnect and connect events.
- Executing the login_procedure database option and verifying the login policy.
- Resetting the connection ID.

Using Connection Pooling with other Connection Pooling Products

If you are using a product or API that supports connection pooling, then the connection pooling of the product or API supersedes SAP Sybase IQ connection pooling. Both types of connection pooling can be active at the same time.

The behavior of connection pooling in your product or the API may be significantly different than SAP Sybase IQ connection pooling. If the behavior of connection pooling in your product or API is inappropriate for an application, SAP Sybase IQ connection pooling can be used and may improve the performance of some applications.

Connection Pooling and Read-Only Scale-Out

If the NodeType (NODE) connection parameter is also specified for a connection, the application typically connects to the primary server and the primary server determines which copy node is least heavily loaded. The connection is then redirected to that node. If the application makes and drops several such connections within a short period of time, the connection is pooled and the primary server is not asked which copy node to use. This behavior reduces the load on the primary server, but may not give expected behavior.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42

- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Temporary Connections

Temporary connections perform operations like running backups or initializing databases.

Use the system procedures **sa_conn_info** or **sa_conn_list** for information about temporary connections. The ParentConnection property returns the connection ID of the connection that spawned the temporary connection.

Temporary connections have connection IDs that are larger than 1 billion (1000000000), and their names describe the function of the connection.

The following example uses the **sa_conn_info** system procedure to return a result set showing which connection created a temporary connection.

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

Connection 8 spawned the temporary connection that executed a **CREATE DATABASE** statement.

```
Number         Name                ParentConnection
-------------------------------------------------------------
1000000048     INT: CreateDB       8
9              SQL_DBC_14675af8    (NULL)
8              SQL_DBA_152d5ac0    (NULL)
```

See also

- *Roadmap for Connections* on page 41

- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Logical Server Configuration

In a multiplex, you can only access servers by using logical servers.

Not all member nodes may be available at all times to its logical server for reasons such as a failure or exclusion of a member node from the multiplex. At any given time, the effective logical server configuration represents the current dynamic constitution of the logical server consisting of all member nodes that are actually available for use to the logical server. The effective logical server configuration is essentially a function of static logical server configuration and dynamic state of the multiplex.

Note: Only multiplex configurations support logical servers. Information about built-in logical servers and logical server policies can remain in the catalog in a simplex environment, but is not used.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55

- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *How to End Connections* on page 105
- *Connection Logging* on page 106

Connections in Simplex

In simplex, connections are unaffected by the login policy setting of logical server assignments.

Connections have no logical server context. The base setting of login policy option ‘locked’ is applied before a server accepts connections.

Connections in Multiplex

In a multiplex, login policies control access to logical servers. All users of a login policy can only access those multiplex servers that are effective members of the assigned logical servers.

When you connect to a multiplex server, logical server context of the connection is determined based on the effective logical server assignment of the user's login policy and the current node.

Logical Server Context of a Connection

When you establish a user connection, the user's login policy and the current node determine the logical server context of the connection.

- When effective logical server assignment for the user’s login policy is one or more logical servers, then the logical server context of the connection is determined based upon current node’s unambiguous membership in one of the specified logical servers.

  Note: Logical servers to which a given login policy allows access do not have overlapping membership.
- A connection fails if the current node is not a member of any of the logical servers assigned to the user’s login policy.
- A connection also fails if the effective logical server assignment for the user’s login policy is set to NONE.

How to End Connections

Connections must be ended under certain circumstances.

Connections end when:

- In Interactive SQL or Embedded SQL, a user or application issues an explicit DISCONNECT statement for the current connection, a specified connection, or all connections for that application
- In Interactive SQL, a user selects SQL > Disconnect
- An application with active connections, such as Interactive SQL, is closed

Users with DROP CONNECTION system privilege can also drop a specific connection in Interactive SQL or Embedded SQL by issuing a DROP CONNECTION statement.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *Connection Logging* on page 106

Connection Logging

By default, each time a user connects or disconnects from a database, the `<dbname>.iqmsg` log records this action.

You can control logging of user connections and disconnections using the database option, LOG_CONNECT. If connection logging is disabled when a user connects, and then turned ON before the user disconnects, the message log shows that user disconnecting but not connecting.

For more information, see *Reference: Statements and Options > Database Options > Alphabetical List of Options > LOG_CONNECT Option*.

See also

- *Roadmap for Connections* on page 41
- *Ways to Connect* on page 42
- *Supported Connection Interfaces* on page 52
- *Connection Status* on page 53
- *How Connection Parameters Work* on page 54
- *Connection Parameters in ODBC Data Sources* on page 55
- *Interactive SQL Connections* on page 56
- *Using Server Classes for Remote Data Access* on page 63
- *ODBC Data Sources* on page 79
- *File Data Sources* on page 85
- *The iAnywhere Solutions 16 Oracle ODBC Driver* on page 87
- *Database Connections Using OLE DB* on page 90
- *Connections from Other Databases* on page 93
- *How to Test Connections* on page 94
- *Integrated Logins* on page 95
- *Connection Pooling* on page 100
- *Temporary Connections* on page 102
- *Logical Server Configuration* on page 103
- *How to End Connections* on page 105
- *LOG_CONNECT Option* on page 412

Create and Manage SAP Sybase IQ Databases

Use Interactive SQL or Sybase Control Center to create and manage SAP Sybase IQ databases.

Create SAP Sybase IQ Databases

To define your database, use SQL statements or a database design tool.

Some application design systems, such as Sybase PowerDesigner®, contain facilities for creating database objects. These tools construct SQL statements that are submitted to the server, typically through its ODBC interface. If you are using one of these tools, you do not need to construct SQL statements to create tables, assign permissions, and so on.

Database design tools such as Sybase PowerDesigner provide a thorough and reliable approach to developing well-designed databases.

Database Creation with Sybase Control Center
To create a SAP Sybase IQ database in Sybase Control Center, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

Database Creation with SQL

You can use SQL statements for defining database objects directly if you build your database from an interactive SQL tool, such as **dbisql**.

Even if you use an application design tool, you may want to use SQL statements to add features to the database if they are not supported by the design tool.

Once the database is created, you can connect to it and build the tables and other objects that you need in the database.

Before You Create a Database

Perform prerequisite actions before creating a database using SQL statements.

In order to create a database using SQL statements, you must:

- Start the database server
- Start **dbisql**

To create a database in **dbisql**, you need to connect to an existing database, or else start the *utility database*, a phantom database with no database files and no data.

Note: If the server is started with the **-m** server option, you cannot create a database.

Starting the Utility Database

You must start the utility database before creating new databases if no databases are built yet.

You can start the utility database in either of these ways:

1. Start the database server without a database by specifying only `-n enginename` on the startup command.
2. Start **dbisql** from the command line, setting the Database Name to `utility_db` in the connection string, as in:

   ```
   dbisql -c "uid=dba;pwd=sql;eng=myserver;dbn=utility_db;...
   ```

   (You must not specify it as the Database File, because utility_db has no database file.)

Dbspaces Created Automatically

When you create a database, the database server creates certain dbspaces automatically.

Dbspace name	Purpose	Default operating system file name
`IQ_SYSTEM_MAIN`	Main (permanent) IQ store file	*dbname.iq*
`IQ_SYSTEM_MSG`	Message log file	*dbname.iqmsg*
`IQ_SYSTEM_TEMP`	Temporary IQ store file	*dbname.iqtmp*
`IQ_SHARED_TEMP`	Temporary IQ store	Initially, this dbspace contains no files.
`SYSTEM`	Catalog store file	*dbname.db*

The `SYSTEM` dbspace contains the system tables, which hold the schema definition as you build your database. It also holds a separate checkpoint log, rollback log, and optionally a write file, transaction log, and transaction log mirror, for the catalog store.

Note: In addition to these database files, the database server also uses a temporary file to hold information needed during a session. This temporary file is not the same as the IQ temporary store, and is not needed once the database server shuts down. The file has a server-generated name with the extension `.tmp`. Its location is determined by the TEMP environment variable, or the coordinator environment variable on UNIX.

Database File Placement

When you create a database, consider whether you will ever need to move the database. The location of files for dbspaces may also affect performance.

The IQ catalog (`.db`) and transaction log (`.log`) files can be safely moved. Never attempt to copy a running database. If you use relative path names to create the database, then you can move the files by shutting down the server and using the operating system copy file command.

If you use absolute (fully qualified) path names to create the database, then you must move the files by using the **BACKUP** command to make a full backup, and the **RESTORE** command with the **RENAME** option to restore the backup.

IQ dbspaces on raw partitions can be moved to other partitions while the database is shut down. The new partition must be at least as large as the current dbspace size. The new partition must also have the same path in order for the dbspace to start.

You can use a byte copy utility such as dd only to copy raw data to a file system device or another raw device. Never use byte copy to move dbfiles from a file system to a raw device.

Warning! When you allocate system files for dbspaces (system, IQ main or IQ temporary), do not place the files on a file system that is shared over a local area network. Doing so can lead to poor I/O performance and other problems, including overloading the local area network and problems in the dbspace file. On UNIX and Linux platforms, avoid Network File System (NFS) mounted file systems. On Windows, do not place dbspace files on network drives owned by another node. These file placement recommendations also apply to log files.

To avoid conflicts, one database administrator on a single connection should manage all dbspaces.

Performance related to randomly accessed files, including the system, IQ main, and IQ temporary dbfiles, can be improved by increasing the number of disk drives devoted to those files. Performance related to sequentially accessed files, including the transaction log and message log files, can be improved by locating these files on dedicated disk drives.

Suggestions to reduce file placement impact on performance:

- Keep random disk I/O away from sequential disk I/O.
- Place the database file, temporary dbspace, and transaction log file on the same physical machine as the database server.
- Isolate SAP Sybase IQ database I/O from I/O for proxy tables in other databases, such as Adaptive Server Enterprise.
- Place the transaction log and message log on separate disks from the IQ store, catalog store, and temporary store, and from any proxy databases such as Adaptive Server Enterprise.

If your IQ requirements are large and complex enough that you need multiple physical systems, consider using SAP Sybase IQ multiplex functionality.

Raw Device Permissions

Make sure that all raw devices have read and write permissions before you create a database or add a dbspace. Check to see that `/dev/rawctl` has read permission. Raw device names on Linux use `/dev/raw/r`*`devname`*. For example, `/dev/raw/raw10`.

Database File Compatibility

SAP Sybase IQ servers cannot manage databases created with versions prior to SAP Sybase IQ 12.7; likewise, old servers cannot manage new databases.

CREATE DATABASE Statement Defaults

The **CREATE DATABASE** statement has two required parameters and several optional parameters.

You must specify the file name for catalog store and the **IQ PATH**. All other parameters are optional.

If you use all of the defaults, your database has these characteristics:

- Case-sensitive (**CASE RESPECT**). “ABC” compares NOT EQUAL to “abc”. The default login is user ID DBA (uppercase) and password sql (lowercase). By default, passwords are case sensitive. User names are always case insensitive.
- Catalog page size of 4096 bytes (**PAGE SIZE 4096**).
- When comparing two character strings of unequal length, IQ treats the shorter one as if it were padded with blanks to the length of the longer one, so that ‘abc’ compares equal to ‘abc’ (**BLANK PADDING ON**).
- Incompatible with Adaptive Server Enterprise.
- IQ page size is 128KB (**IQ PAGE SIZE 131072**).
- IQ message file and IQ temporary store are in the same directory as the catalog store.
- For a raw device, **IQ SIZE** and **TEMPORARY SIZE** are the maximum size of the raw partition. For operating system files, see the discussion of this parameter below.
- IQ temporary store size is half the IQ size.
- jConnect JDBC driver is enabled (**JCONNECT ON**).
- The collation ISO_BINENG is used. The collation order is the same as the order of characters in the ASCII character set. In a case-sensitive database, all uppercase letters precede all lowercase letters (for example, both 'A' and 'B' precede 'a').
- **IQ RESERVE** and **TEMPORARY RESERVE** are 0.
- **SYSTEM PROCEDURE AS DEFINER** is OFF.

Increase Password Security

Passwords are an important part of any database security system. There are several options available to increase password security.

Implement a Login Policy

Use a login policy to control the frequency of password changes, to specify the number of login attempts allowed before an account is locked, or to force password expiration. See *Login Policies*.

Implement a Minimum Password Length

By default, passwords can be any length. For greater security, you can enforce a minimum length requirement on all new passwords to disallow short (and therefore easily guessed) passwords. The recommended minimum length is 6. See *MIN_PASSWORD_LENGTH*.

Implement Password Rules
You can implement advanced password rules that include requiring certain types of characters in the password, disallowing password reuse, and expiring passwords. Validation of the rules occurs when a new user ID is created or a password is changed. See *VERIFY_PASSWORD_FUNCTION*.

See also

- *MIN_PASSWORD_LENGTH Option* on page 413
- *VERIFY_PASSWORD_FUNCTION Option* on page 413

Relative Path Names

You can create a database using a relative or fully qualified path name for each of the files for the database.

Create databases with relative path names. If you specify absolute path names, you will not be able to move files to a different path name without backing up and restoring the database.

If your database is on UNIX, you can define a symbolic link for each path name.

If you omit the directory path, SAP Sybase IQ locates the files as follows:

- The catalog store 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.
- The transaction log is created in the same directory as the catalog store. (This also occurs if you do not specify any file name.) However, you should place it on a different physical device from the catalog store and IQ store, on the same physical machine.

Note: You must start the database server from the directory where the database is located, for any database created with a relative path name. Using a configuration file to start the server ensures that you start the server from a consistent location.

IQ PATH Parameter Guidelines

The required **IQ PATH** parameter tells SAP Sybase IQ that you are creating an IQ database, not a SQL Anywhere database.

You specify the location of your IQ store in this parameter.

Choose a location for your database carefully. Although you can move an IQ database or any of its files to another location, to do so, you must shut down the database and you may have to perform a backup and restore.

You can add space on a different drive, but you can only use this additional space for new data. You cannot readily move a particular index, table, or rows of data from one location to another.

Each operating system has its own format for raw device names. See *Reference: Building Blocks, Tables, and Procedures > Physical Limitations* for an important note about initializing raw devices on Sun Solaris.

Table 3. Raw Device Names on UNIX

UNIX Platform	Example
AIX	`/dev/rraw121v`
HP-UX	`/dev/vg03/rrchee12g`
Sun Solaris	`/dev/rsd0c`
Sun AMD	`/dev/rdsk/c5t0d0s1`

Table 4. Raw Device Names on Windows

Device type	Name format required	SQL example
Partitioned	Letter assigned to that partition	`\\\\.\\C:`
Not partitioned	*PhysicalDriveN*, where *N* is a number starting with 0 and going as large as needed. You can find the physical drive numbers by running Disk Administrator in Administrative Tools.	`\\\\.\\ PhysicalDrive32`

On Windows systems, when you specify device names that include a backslash, you must double the backslash to keep the system from mistaking a backslash/letter combination for an escape sequence such as tab or newline command.

You must always double the backslash when naming raw devices on Windows in SQL statements.

Example 1

The following statement creates an IQ database called `company.db`. This database consists of four Windows files:

- The catalog store is in `company.db`, in the directory where the server was started (in this case, `c:\company`)
- The IQ store is in `c:\company\iqdata\company.iq`
- The temporary store is in `c:\company\company.iqtmp`
- The IQ message log file is in `c:\company\company.iqmsg`

```
CREATE DATABASE 'company.db'
IQ SIZE 200
IQ PATH 'c:\\company\\iqdata\\company.iq'
```

Example 2
The following statement creates an IQ database called `company.db`. This database consists of four UNIX files:

- The catalog store is in `company.db`, in the directory where the server was started (in this case, `/disk1/company`)
- The IQ store is in `/disk1/company/iqdata/company.iq`
- The temporary store is in `/disk1/company/company.iqtmp`
- The IQ message log file is in `/disk1/company/company.iqmsg`

```
CREATE DATABASE 'company.db'
IQ SIZE 2000
IQ PATH '/disk1/company/iqdata/company.iq'
```

Example 3
The following UNIX example creates an IQ database called `company` with a raw partition for IQ PATH.

```
CREATE DATABASE 'company'
IQ PATH '/dev/rdsk/c0t0d0s0'
```

Example 4
The following Windows example creates an IQ database called `company` with a raw partition for IQ PATH.

```
CREATE DATABASE 'company'
IQ PATH '\\\\.\\D:'
```

IQ PAGE SIZE Parameter Guidelines

The IQ PAGE SIZE parameter determines memory and disk use.

You set a page size for the IQ store with the **IQ PAGE SIZE** option. The **IQ PAGE SIZE** must be a power of 2, from 65536 to 524288 bytes. The IQ page size is the same for all dbspaces in the IQ store.

To obtain the best performance, use the following minimum IQ 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 may 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 may grow to a total size of 8TB or greater.

Multiuser environments, and systems with memory constraints, both benefit from an IQ page size of at least 64KB, as this size minimizes paging.

SAP Sybase IQ stores data on disk in compressed form. It uncompresses the data and moves data pages into memory for processing. The IQ page size determines the amount of disk compression and the default I/O transfer block size for the IQ store. For most applications, this default value is best. For information on these settings and other options that affect resource use and performance, see *Performance and Tuning > Manage System Resources*.

Page Size for Wide Data

If your database includes very wide tables, you may find that the next higher IQ page size for a given number of rows gives you better performance. For example, tables with multiple columns of wide `CHAR` or `VARCHAR` data (columns from 255 to 32,767 bytes) are likely to need a larger than usual IQ page size.

Because IQ stores data in columns, it does not have a true maximum row length. The practical limit, however, is half your IQ page size, because that is the widest result set that a query is guaranteed to be able to return to the client. Choose an IQ page size at least twice the width of the widest table possible.

Database Size Guidelines

When you create a database, you set the size and reserve size in MB of the initial IQ database file (the `IQ_SYSTEM_MAIN` dbspace).

These values are defined in the **IQ SIZE** and **IQ RESERVE** parameters for the main store and **TEMPORARY SIZE** and **TEMPORARY RESERVE** for the temporary store.

- For raw partitions, you do not need to specify **IQ SIZE** or **TEMPORARY SIZE**; SAP Sybase IQ determines the size of the raw devices and sets **IQ SIZE** and **TEMPORARY SIZE** automatically. If you do specify size, the size cannot be larger than the actual raw partition size.
- For operating system files you can rely on the defaults or specify a value based on the size of your data, from the required minimums up to a maximum of 4TB, in 1MB increments.

Default and Minimum Sizes of IQ and Temporary Stores

The **IQ RESERVE** and **TEMPORARY RESERVE** parameters reserve a range of blocks, so that the dbspace can be resized at a later time. Making **IQ RESERVE** larger than needed can use additional disk space, however.

Table 5. Default and Minimum Sizes of IQ and Temporary Stores

IQ page size	Default size of IQ store	Default size of temporary store	Minimum IQ store size when specified explicitly	Minimum temporary store size when specified explicitly
65536	4096000	2048000	4MB	2MB
131072	8192000	4096000	8MB	4MB

IQ page size	Default size of IQ store	Default size of temporary store	Minimum IQ store size when specified explicitly	Minimum temporary store size when specified explicitly
262144	16384000	8192000	16MB	8MB
524288	32768000	16384000	32MB	16MB

PAGE SIZE Parameter Guidelines

Set the catalog store page size with the **CREATE DATABASE PAGE SIZE** option. The default and minimum value for this option is 4096 (4KB).

Example

The following statement creates a database with a catalog **PAGE SIZE** of 4096 bytes (4KB), where the IQ store is on a UNIX raw partition and has an **IQ PAGE SIZE** of 131072 bytes (128KB). By default, the IQ store size is the size of the raw partition and the temporary store is half that size. Because no path is specified for the temporary store, it is created in the same directory as the catalog store.

```
CREATE DATABASE 'company'
IQ PATH '/dev/rdsk/c2t6d0s3'
PAGE SIZE 4096
IQ PAGE SIZE 131072
```

Block Size Guidelines

In nearly all cases you should rely on the default block size, which is based on the IQ page size.

IQ Main Store and IQ Temporary Store Space Management

Options **MAIN_RESERVED_DBSPACE_MB** and **TEMP_RESERVED_DBSPACE_MB** provide room for checkpoint, commit, and release savepoint operations.

These options determine the reserve space allocation size in the last readwrite dbfile in IQ_SYSTEM_MAIN or IQ_SYSTEM_TEMP, respectively.

The user with the SET ANY PUBLIC OPTION system privilege can limit the amount of space used per connection. In addition, when SAP Sybase IQ runs out of space in IQ main store or the IQ temporary store, the server no longer suspends the transaction that ran out of space until new space is added. The transaction that runs out of space in the IQ main store or the IQ temporary store fails and is rolled back.

The database option `MAX_TEMP_SPACE_PER_CONNECTION` limits the amount of IQ temporary store space used per connection and tracks temporary store usage for all Data Manipulation Language (DML) statements, in addition to queries.
`MAX_TEMP_SPACE_PER_CONNECTION` monitors and limits the actual run time

temporary store usage by the statement. If the connection exceeds the quota set by the MAX_TEMP_SPACE_PER_CONNECTION option, an error is returned and the current statement rolls back.

The default value of the QUERY_TEMP_SPACE_LIMIT database option is 0, which means there is no limit on temporary store usage by queries. To limit the temporary store usage per connection, the DBA can set the MAX_TEMP_SPACE_PER_CONNECTION option for all DML statements, including queries.

When an SAP Sybase IQ database is upgraded from a release prior to version 15.0, the MAX_TEMP_SPACE_PER_CONNECTION database option is set to the default value of 0. You can use **sp_iqcheckoptions** to find the default and current values of options before and after upgrading, to help determine if the new option settings are appropriate for the upgraded database.

Dropping a Database

Dropping a database deletes all tables and data from disk, including the transaction log that records alterations to the database. It also drops all of the dbspaces associated with the database.
To drop a database, use the following SQL statement:

```
DROP DATABASE dbname
```

You must specify the database name and its path name exactly as they were specified when the database was created.

For example, on a Windows system:

```
DROP DATABASE 'c:\sybase\data\mydb.db'
```

The database must be stopped before you can drop it. If the connection parameter **AUTOSTOP=no** is used, you may need to issue a **STOP DATABASE** statement.

Disconnecting Other Users From a Database

SAP Sybase IQ gives you the ability to disconnect other users from a given database.

You can obtain the *connection-id* for a user by using the **connection_property** function to request the connection number. The following statement returns the connection ID of the current connection:

```
SELECT connection_property( 'number' )
```

1. Connect to an existing database as a user with administrative authority over the DROP CONNECTION system privilege.
2. Using Interactive SQL, execute a **DROP CONNECTION** statement.

   The following statement drops the connection with ID number 4.

```
DROP CONNECTION 4
```

Disconnecting from a Database in Embedded SQL

When you are finished working with a database, you can disconnect a named connection or all connections.
Execute an **EXEC SQL DISCONNECT** statement.

The following statement shows how to use DISCONNECT in Embedded SQL:

```
EXEC SQL DISCONNECT :conn_name
```

Disconnecting All Connections from a Database in Interactive SQL

When you are finished working with a database, you can disconnect a named connection or all connections.
Execute a **DISCONNECT** statement.

The following statement shows how to use **DISCONNECT** from Interactive SQL to disconnect all connections:

```
DISCONNECT ALL
```

Showing System Objects in Interactive SQL

In Interactive SQL, you cannot query system tables, but you can browse the contents of a system view.

Most system tables have equivalent system views that you can query.

In a database, a table, view, stored procedure, or domain is a system object. System tables store information about the database itself, while system procedures, and domains largely support Sybase Transact-SQL compatibility.

1. Connect to a database using Interactive SQL.
2. Execute a **SELECT** statement, specifying the system view for the table you want to browse.

   To browse the `ISYSTAB` system table, show the contents of the view `SYS.SYSTAB` in the Results pane.

   ```
   SELECT *
   FROM SYS.SYSTAB
   ```

Setting Database Options in Interactive SQL

Database options are configurable settings that change the way the database behaves or performs.
Specify the properties within a SET OPTION statement.

Note: When you set options for the database itself, you are actually setting options for the PUBLIC role in that database, because all users and roles inherit option settings from PUBLIC.

Manage Data Storage

Specify how SAP Sybase IQ allocates disk space and make sure that data is organized on disk for optimal performance.

Space Allocation

All SAP Sybase IQ databases are preallocated, whether they reside in a file system or a raw partition.

Each database includes multiple tablespaces. A *tablespace* is a unit of storage within the database that you can administer as a logical subset of total storage. You can allocate individual objects and subobjects to individual tablespaces.

A *dbspace* is a tablespace that consists of one or more operating system files.

A *store* is one or more dbspaces that store persistent or temporary data for a special purpose. SAP Sybase IQ has five stores:

- The catalog store contains the SYSTEM dbspace and additional user-defined catalog dbspaces.
- The IQ main store contains the IQ_SYSTEM_MAIN dbspace and other user dbspaces.
- The IQ temporary store contains the IQ_SYSTEM_TEMP dbspace.
- The IQ shared temporary store contains the IQ_SHARED_TEMP dbspace.
- The row-level versioning (RLV) store is an in-memory store for high-performance row-level updates.

Types of Dbspaces

Each type of dbspace stores a particular type of SAP Sybase IQ data.

Dbspace Type	Data Stored	Files Contained by Dbspace	Number of Dbspaces
SYSTEM dbspace	System tables, views, stored procedures, SQL Anywhere tables, and function definitions.	One	One or more
Other catalog dbspaces	SQL Anywhere tables.	One	One or more

Dbspace Type	Data Stored	Files Contained by Dbspace	Number of Dbspaces
IQ_SYSTEM_MAIN	IQ database structures including IQ rollforward and rollback data for each committed transaction and each active checkpointed transaction, the incremental backup metadata, and database space and identity metadata. You may store IQ user objects here, but it is better to store them in other main dbspaces.	One or more	One
Other main dbspaces (also called user dbspaces)	IQ objects such as tables, indexes, and table metadata.	One or more	One or more
RLV_STORE	Real-time in-memory data from row-level versioning (RLV)-enabled tables.	One	One
IQ_SYSTEM_TEMP	Set of 1 to n temporary dbfiles that defines a single temporary dbspace for a standalone database or multiplex node. See the following table for details.	One or more	One
IQ_SHARED_TEMP	Set of 1 to n temporary dbfiles that define a single temporary dbspace shared by all multiplex nodes. See the following table for details.	One or more (initially has no files)	One
IQ_SYSTEM_MSG	External file that logs messages about database activity.	One per multiplex node	One

The dbspace of a table is implicitly or explicitly specified. For base tables, the value of the **DEFAULT_DBSPACE** option implicitly determines the dbspace location, or you can explicitly specify the location using the **CREATE TABLE IN** *dbspace_name* clause. Typically, create base tables in a dbspace in the IQ main store, but you can also create them without IQ indexes in a dbspace in the catalog store.

For global temporary tables, specify the **IN SYSTEM** clause to explicitly create an SA global temporary table. By default, IQ temporary tables are created in IQ_SYSTEM_TEMP.

Always Stored in IQ_SYSTEM_TEMP	Always Stored in IQ_SHARED_TEMP	May Be Stored in Either IQ_SHARED_TEMP or IQ_SYSTEM_TEMP
Scratch space for recovery from server failure	Distributed temporary objects for distributed query processing	IQ temporary user objects such as IQ temporary tables and indexes
Local bitmaps for shared storage space management		Scratch space for local, nonversioning temporary objects
Temporary storage transaction state (savepoint roll forward and roll back)		

Catalog Store

The catalog store contains the *metadata* for the IQ database.

Metadata describes the layout of IQ tables, columns, and indexes. The catalog store is sometimes referred to simply as the catalog.

- The SYSTEM dbspace – IQ catalog dbspace SYSTEM contains metadata for your IQ database, stored in the same format as tables in a SQL Anywhere relational database system. SQL Anywhere can exist with or without IQ. You may have SQL Anywhere-style tables in your catalog store along with your IQ tables, or you may have a separate SQL Anywhere database. Each catalog dbspace contains exactly one file.
- Other catalog dbspaces – you can create SQL Anywhere tables in a separate dbspace from the SYSTEM dbspace.

You can preallocate space to your catalog store by executing an **ALTER DBSPACE** statement.

IQ_SYSTEM_MAIN Dbspace

IQ_SYSTEM_MAIN is a special dbspace that contains structures necessary for the database to open: the IQ checkpoint log, IQ rollforward/rollback data for each committed transaction and each active checkpointed transaction, the incremental backup metadata, and database space and identity metadata.

The IQ_SYSTEM_MAIN dbspace is created at database creation or when you upgrade an older IQ database to SAP Sybase IQ 16.0. IQ_SYSTEM_MAIN is always online when the database is open.

Other User Main Dbspaces

Create user main dbspaces so that users do not place user tables or indexes in IQ_SYSTEM_MAIN.

The best practice is to avoid placing user tables or indexes in IQ_SYSTEM_MAIN. The administrator may allow user tables to be created in IQ_SYSTEM_MAIN, especially if these tables are small, very important tables. However, the recommended method is that immediately after creating the database, the administrator creates a second main dbspace (a user main dbspace), revokes CREATE privilege in dbspace IQ_SYSTEM_MAIN from PUBLIC, grants CREATE privilege for the new main dbspace to selected users or PUBLIC, and sets PUBLIC.DEFAULT_DBSPACE to the new user 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';
```

IQ Temporary Dbspace

A single dbspace for the IQ temporary store, IQ_SYSTEM_TEMP, is created when you create a database or upgrade an older IQ database.

Each IQ dbspace may contain any number of files. The only limit is that the total number of IQ files is 16383.

Ensure that the IQ_SYSTEM_TEMP dbspace is at least as large as the value specified in the **-iqtc** server option of the **start_iq** utility.

IQ Shared Temporary Dbspace

A single dbspace for the IQ shared temporary store, IQ_SHARED_TEMP, is created when you create a database or upgrade a 12.7 ESD #5 or higher IQ database to SAP Sybase IQ 16.0. This dbspace stores temporary structures that are shared among nodes for distributed query processing.

Initially, this dbspace contains no files. To add files, use **ALTER DBSPACE ADD FILE**. Allocating files to this dbspace is optional and required only for distributed query processing in multiplex servers.

When you set the `TEMP_DATA_IN_SHARED_TEMP` logical server policy option ON, all temporary table data and eligible scratch data writes to the shared temporary store, provided that the shared temporary store is not empty. You must restart secondary 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 secondary nodes, data

instead writes to IQ_SYSTEM_TEMP. When OFF, all temporary table data and scratch data writes to the local temporary store.

See also

- *ALTER LS POLICY Statement* on page 424

Shared System Temporary Store

A multiplex configuration with shared temporary storage can use the IQ_SHARED_TEMP dbspace as a shared system temporary store instead of requiring a separate local store for each secondary server. The shared system temporary store simplifies multiplex configuration, improves performance, and supports distributed query processing.

On multiplex systems:

- When you set the logical server policy option TEMP_DATA_IN_SHARED_TEMP ON, SAP Sybase IQ creates all temporary objects on the IQ_SHARED_TEMP dbspace. You must restart secondary 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 secondary nodes, data instead writes to IQ_SYSTEM_TEMP.)
- Temporary user objects (such as tables or table indexes) that you create using the **IN IQ_SYSTEM_TEMP** clause go in either IQ_SYSTEM_TEMP or IQ_SHARED_TEMP, depending on the value of the logical server option TEMP_DATA_IN_SHARED_TEMP:
  - If TEMP_DATA_IN_SHARED_TEMP is 'OFF', objects go in IQ_SYSTEM_TEMP.
  - If TEMP_DATA_IN_SHARED_TEMP is set 'ON', objects go in IQ_SHARED_TEMP.

  SAP Sybase IQ does not support creating temporary user objects using the **IN IQ_SHARED_TEMP** clause.
- The **WITH STOP SERVER** clause automatically shuts down all servers in the logical server. These statements support **WITH STOP SERVER**:
  - **ALTER LOGICAL SERVER**
  - **ALTER LS POLICY**
  - **CREATE LOGICAL SERVER**
  - **DROP LOGICAL SERVER**
- If you use **ALTER LS POLICY ... WITH STOP SERVER** to change the TEMP_DATA_IN_SHARED_TEMP option 'ON|OFF', all servers in that logical server shut down automatically. You must restart the servers to force the logical server to place temporary data in the store specified by the TEMP_DATA_IN_SHARED_TEMP option.
- If you use **ALTER LS POLICY** to set TEMP_DATA_IN_SHARED_TEMP 'OFF', the logical server starts placing temporary data in the SYSTEM temporary area after the next normal server startup.
- You can also change the TEMP_DATA_IN_SHARED_TEMP value indirectly using **CREATE LOGICAL SERVER**, **ALTER LOGICAL SERVER**, or **DROP LOGICAL SERVER** statements and the **WITH STOP SERVER** clause.

See also

- *ALTER LS POLICY Statement* on page 424
- *CREATE LS POLICY Statement* on page 465
- *CREATE TABLE Statement* on page 467

IQ Message File Dbspace

IQ_SYSTEM_MSG is a system dbspace that points to the file path of the database IQ message log file.

IQ_SYSTEM_MSG is not considered a store because it does not store any data. It has one file per multiplex node. By default, the physical file name for the message file on a simplex server or a coordinator of a multiplex is `<dbname>.iqmsg`. The physical file name for the IQ message file on a secondary node in a multiplex is `<servername>.iqmsg`.

Because it is not an IQ store dbspace, **ALTER** commands such as **READONLY** and **OFFLINE** do not apply to IQ_SYSTEM_MSG.

Space for Databases

The first dbspace for each store is created automatically when you create the database. You can create additional dbspaces as needed.

When you create and load a table, SAP Sybase IQ distributes data among all existing dbspaces in that store with available space. You can reserve space for a dbspace to grow when you create it. You can resize the dbspace up to the maximum reserve. You can also make the dbspace smaller, provided that all data has been moved off of the truncated portion of the dbspace. You can move individual database objects off of specified dbspaces as needed.

Do not allocate all of your disk space to your IQ database. Keep ten percent in reserve. SAP Sybase IQ needs this space to gracefully handle out-of-space conditions.

Create all dbspaces when you create the database, rather than adding them gradually as old ones fill. This ensures that your dbspaces are filled more evenly, improving disk I/O.

Create separate databases for debugging purposes. Avoid performing development work on production databases, because it increases the possibility of a server failure.

Space Requirements for IQ Stores and Temporary Stores

The amount of data, and the number and types of indexes you create, determine how much space you need in your IQ database.

If you run out of space when loading or inserting into a database, the IQ server rolls back either the entire transaction or rolls back to a savepoint.

In addition to any temporary tables you explicitly define, SAP Sybase IQ uses the temporary store as a temporary result space for sorts, hashes, and bitmaps during loads and deletions. The types of queries issued, the degree of concurrent use, and the size of your data all determine how much space you need for your temporary store.

Sizing Guidelines for Main and Temporary Stores

The SAP Sybase IQ architecture influences guidelines for data storage.

- The IQ_SYSTEM_MAIN dbspace holds all of the database metadata other than IQ table metadata. IQ table metadata is stored in the table's dbspace and the table version (TLV) log. If a node is down, the multiplex must store versions to synchronize them when the node comes back up. These versions may use large amounts of space.
- Approximately 20 percent of the IQ_SYSTEM_MAIN dbspace is now used for preallocated free list space and not available for user data.
- Because this version of SAP Sybase IQ performs more operations in parallel, it uses more temporary space than earlier versions.

Three factors influence the space required for the IQ_SYSTEM_MAIN store:

- Versioning – the volume of versions maintained varies.
- Nature of data and indexes.
- Dynamic nature of the data – the capacity to load more data at any time.

While documentation can offer general guidelines, the combination of these factors makes each database's requirements unique. For a development or report server with a total size under 500GB, an IQ_SYSTEM_MAIN file of 10 to 20GB may suffice.

Table 6. Size Guidelines for IQ_SYSTEM_MAIN and IQ_SYSTEM_TEMP in Production Databases

Task	Guideline	Notes
Loading empty schema from **iqunload -n** output or for a small test database	10GB main, 5GB temporary	CREATE DATABASE sizes are in MB. The server must be at 12.7 ESD #5 or higher to use **iqunload -n**.

Task	Guideline	Notes
Creating new production database	• If you are migrating a database, and use a raw device for your current IQ_SYSTEM_MAIN, assign a new unused raw device of your standard size. • Total size of IQ_SYSTEM_MAIN should be at least 1/100 total database size, with a minimum 100GB main and 100GB reserve. • If using raw disks for IQ_SYSTEM_MAIN, use multiple raw disks whenever possible. Multiple raw disks enable SAP Sybase IQ to stripe the data across devices, which improves performance. • For IQ dbspaces in production, use a fault-tolerant file system implemented by a high-performance, redundant disk array (for example, RAID 5). For single-server systems, you can use a local file system, but multiplex systems require raw devices, ideally on a Storage Area Network device.	Omit ms_size if specifying a raw device. On a Windows system, only user accounts with Administrator privilege can access raw devices. The **rawaccedit** utility sets permission for devices for the current session. Always set the main reserve to 20 percent of IQ_SYSTEM_MAIN size. To set the main reserve, use the database option `MAIN_RESERVED_DBSPACE_MB`.
Creating main store for a multiplex	Double the space recommended for a simplex database, or at least 200GB main and 200GB reserve dbspace.	

Example 1

In **CREATE DATABASE** syntax, default size units are in MB, not GB.

The following statement creates a database with 100GB IQ_SYSTEM_MAIN with 100GB reserve (for future expansion):

```
CREATE DATABASE 'test.db'
IQ PATH 'test.iq'
IQ SIZE 100000
IQ RESERVE 100000
TEMPORARY PATH 'test.iqtmp'
TEMPORARY SIZE 5000
```

Example 2

`MAIN_RESERVED_DBSPACE_MB` lets you control the amount of space SAP Sybase IQ sets aside in your IQ main store for certain small but critical data structures used during release savepoint, commit, and checkpoint operations.

Set the `MAIN_RESERVED_DBSPACE_MB` option value to 20 percent of the IQ_SYSTEM_MAIN SIZE. For example, if IQ_SYSTEM_MAIN is 100GB, set it to 20GB, as follows:

```
SET OPTION PUBLIC.MAIN_RESERVED_DBSPACE_MB = 20000
```

Example 3
Specify the IQ_SYSTEM_MAIN size in the database migration command.

The **-ms_size** parameter requires a value in MB, not GB. Omit **-ms_size** if specifying a raw device. For a raw device, you must specify an unused raw partition.

Create an IQ_SYSTEM_MAIN on a raw device on UNIX:

```
iqunload -au -ms_filename /dev/rdsk/c1t0d1 -c
"UID=DBA;PWD=SQL;DBF=latest.db"
```

Create an IQ_SYSTEM_MAIN on a raw device on Windows:

```
iqunload -au -ms_filename \\\\.\\PhysicalDrive1 -c
"UID=DBA;PWD=SQL;DBF=latest.db"
```

Setting Up Windows Access to Raw Devices

Windows systems restrict raw device access to user accounts with Administrator privilege.

To run the SAP Sybase IQ servers using an account that lacks Administrator privilege, you must enable new device access permissions for that account after each system reboot.

The **rawaccedit** utility sets permissions for devices for the current session.

Set up read-write access for the write servers and read access for query servers.

1. Type the following at a command prompt:
   ```
   rawaccedit
   ```
2. In the IQ Raw Device Access window, type the name of the user and the device to which you want to grant access.

   You can use Alt+N to tab to the User's Name box and Alt+D to tab to the Raw Device Name box.

To specify...	Type...
An unpartitioned raw device	Type the physical drive number. Unpartitioned drives are named `\\.\PhysicalDriveN`, where N is a number starting with 0. To find the physical drive numbers, Run **Accessories > System Tools > System Information**.
A partitioned raw device	Type the letter assigned to that partition.

3. Click **Add**.

4. Correct any errors in the user name and device name that display in the top panel and click **Update ACL and Exit**.

Device access permissions remain until you reboot Windows.

Viewing Access Permissions of a RAW Device

Display the current access permissions of a Windows RAW device.

The **rawaccess** utility must be run as an admin user. This is not the same as having admin privileges. This can be done by right-clicking **Command Prompt** and selecting **Run As Administrator**. "Administrator:" appears in the window title bar when running as an administrator.

The permission_types comply with the Microsoft Access Control Entry (ACE) for Windows. For more information please see *http://msdn.microsoft.com/en-us/library/windows/desktop/aa374899(v=vs.85).aspx*

1. Open the Command prompt window running as an administrator.
2. Type **rawaccess** followed by the RAW device letter:

   ```
   rawaccess filename
   ```

   Where:
   - **filename –** the Windows universal namespace filename path to the RAW device in one of the two formats:
     - \\.*logical_drive_name*
     - \\.*physical_device_id*
   - **logical_drive_name –** the partitioned letter used to represent the disk partition (for example: \\.\D:).
   - **physical_device_id –** the name assigned by the Window device manager. These names can be looked up using the Disk Manager in the control panel (for example: \\.\PhysicalDisk1).

```
1.  Allow Exec  to \Everyone (Well Known Group)
2.  Allow All   to NT AUTHORITY\SYSTEM (Well Known Group)
3.  Allow All   to BUILTIN\Administrators (Alias)
4.  Allow Exec  to NT AUTHORITY\RESTRICTED (Well Known Group)
```

Setting Permissions to a RAW Device

Set access permissions to a RAW device on Windows is required before you can use it.

To use RAW devices with SAP Sybase IQ, the account running SAP Sybase IQ cannot use UAC (User Account Controls).

The **rawaccess** utility must be run as an admin user. This is not the same as having admin privileges. This can be done by right-clicking **Command Prompt** and selecting **Run As**

Administrator. "Administrator:" appears in the window title bar when running as an administrator.

By default, Windows does not allow direct write access to RAW devices, the /GRANT option is the simplest method to add access as it is merged in with the other default entries setup by the operating system.

The permission_types comply with the Microsoft Access Control Entry (ACE) for Windows. For more information please see *http://msdn.microsoft.com/en-us/library/windows/desktop/aa374899(v=vs.85).aspx*

rawaccess uses the standard Microsoft universal naming conventions for Win32 device namespaces. For additional information see *http://msdn.microsoft.com/en-ca/library/windows/desktop/aa365247(v=vs.85).aspx.*

1. Open the Command prompt window running as an administrator.
2. Type the following, specifying the user to be granted permissions:

   ```
   rawaccess filename [{/permission_type} trustee]
   ```

   Where:
   - **filename –** the Windows universal namespace filename path to the RAW device in one of the two formats:
     - \\.*logical_drive_name*
     - \\.*physical_device_id*
   - **logical_drive_name –** the partitioned letter used to represent the disk partition (for example: \\.\D:).
   - **physical_device_id –** the name assigned by the Window device manager. These names can be looked up using the Disk Manager in the control panel (for example: \\.\PhysicalDisk1).
   - **permission_type –** optional type of access action. If not specified, the current access list appears
     - DENY – Adds an access denied entry to the specified device for the specified trustee.
     - GRANT – Adds an access allowed entry to the specified device for the specified trustees. The access allowed entry is merged with any existing inherited access allowed rights.
     - REVOKE – Removes an access allowed entry to the specified device for the specified trustee.
     - SET – Sets an access allowed entry to the specified device for the specified trustee.
   - **trustee –** a valid login ID for a user account.
3. View the access permissions to verify the permission is correctly set.

Example:

Grant access permission to the RAW device `E:` to `user1`.

```
rawaccess \\.\E: /Grant user1@company.com
```

The access permissions for the RAW device now appear as:

```
1. Allow All   to COMPANY\user1 (User)
2. Allow Exec  to \Everyone (Well Known Group)
3. Allow All   to NT AUTHORITY\SYSTEM (Well Known Group)
4. Allow All   to BUILTIN\Administrators (Alias)
5. Allow Exec  to NT AUTHORITY\RESTRICTED (Well Known Group)
```

Setting Up Symbolic Links for Raw Devices on Windows

Symbolic links simplify raw device access on Windows systems and persist across reboots.

Prerequisites

- Windows Vista or later Windows operating system.
- You must have Administrator privileges in Windows Vista or change the behavior in the Local Security Policy management console.

Task

1. Create a mount point for the symbolic links as an NTFS file system designated by a drive letter, for example, T.
2. At the Windows command prompt, use the **mklink** command to create the symbolic link, for example:

   ```
   mklink T:\IQ_SYSTEM_MAIN \\.\Volume{GUID}
   ```

Scripts that create the database or add space can reference `T:\IQ_SYSTEM_MAIN` instead of the format `\\.\PhysicalDeviceN`.

If storage goes offline, the symbolic link persists when the system comes back online.

Estimating Space and Dbspaces Required

To avoid difficulties when a database or a particular dbspace is full, estimate dbspace requirements before you create the database and the objects in it.

You can run SAP Sybase IQ stored procedures to estimate how much space and how many dbspaces your databases will require.

1. Run **sp_iqestspace** to estimate the amount of space you will need to create a database, based on the number of rows in the underlying database tables. For each table that you plan to create: **sp_iqestspace** *table_name*, *rows*[, *iqpagesize*]

   The amount of space needed by each table is returned as “RAW DATA index_size”.

2. Run **sp_iqestdbspaces** to determine how many dbspace files to create from the given space and what size they should be. Use the total index sizes calculated in step number 5 as the *minsize* and *maxsize* parameters:

```
sp_iqestdbspaces (dbsize [,iqpagesize]
  [,minsize] [,maxsize] ...
```

All these calculations are estimates. Results vary based on the columns and indexes you create for your database. See *Reference: Building Blocks, Tables, and Procedures > System Procedures* for syntax and usage notes for each procedure.

Dbspace Management Example

This section illustrates the dbspace management process from creating a new database and adding objects and data to the database, through relocating objects and dropping the empty dbspace. This example includes sample SQL code and the output of the related system stored procedures.

Creating the Database Objects

Create a small database, `dbspacedb`, using:

```
CREATE DATABASE 'D:\IQ\dbspacedb'
     IQ PATH 'D:\IQ\dbspacedb.iq'
     IQ SIZE 10
     IQ RESERVE 100
     TEMPORARY SIZE 10
     TEMPORARY RESERVE 10
     JCONNECT OFF;
```

Connect to the `dbspacedb` database:

```
CONNECT DATABASE dbspacedb
     user DBA identified by sql;
```

Add two dbspaces to the `dbspacedb` database:

```
CREATE DBSPACE dbspacedb2
USING FILE dbspace2 'D:\IQ\dbspacedb.iq2'
SIZE 10 RESERVE 20MB;CREATE DBSPACE dbspacedb3
USING FILE dbspace3 'D:\IQ\dbspacedb.iq3'
SIZE 10 RESERVE 40MB;
```

Changing the Size of a Dbspace

The main store in the preceding example is too small for the tables to be added in the next example. The **ALTER DBSPACE** commands in this section change the dbspace size.

The database `dbspacedb` has a reserve size of 100MB for the IQ main store, which was set using the **IQ RESERVE** parameter of the **CREATE DATABASE** statement. This IQ main store (the `IQ_SYSTEM_MAIN` dbspace) can be extended by 100MB. The original `IQ_SYSTEM_MAIN` is created with a size of 10 MB (the **IQ SIZE** parameter of **CREATE**

DATABASE). The following **ALTER DBSPACE** command with the **ADD** parameter extends the IQ_SYSTEM_MAIN dbspace by 10MB to 20MB:

```
ALTER DBSPACE IQ_SYSTEM_MAIN ADD 10mb;

sp_iqdbspace;
```

DBSpaceName	DBSpaceType	Writable	Online
IQ_SYSTEM_MAIN	MAIN	T	T
IQ_SYSTEM_TEMP	TEMPORARY	T	T
dbspacedb2	MAIN	T	T
dbspacedb3	MAIN	T	T

Usage	TotalSize	Reserve	NumFiles	NumRWFiles
25	20M	90M	1	1
7	10M	10M	1	1
1	10N	20M	1	1
1	10M	40M	1	1

Stripingon	StripeSize	BlkTypes	OK ToDrop
T	1K	1H,1248F,32D, 177A,128M	N
T	1K	1H,64F,16A	N
T	1K	1H	Y
T	1K	1H	Y

If you do not create the dbspacedb database with an **IQ RESERVE** value, you cannot extend the dbspace. You can, however make the dbspace smaller, and the size taken away from the dbspace is added to the reserve.

The IQ_SYSTEM_MAIN dbspace is now 20MB in size. Resize the dbspace to 15MB:

```
ALTER DBSPACE IQ_SYSTEM_MAIN SIZE 15mb;

sp_iqdbspace;
```

DBSpaceName	DBSpaceType	Writable	Online
IQ_SYSTEM_MAIN	MAIN	T	T
IQ_SYSTEM_TEMP	TEMPORARY	T	T
dbspacedb2	MAIN	T	T
dbspacedb3	MAIN	T	T

Usage	TotalSize	Reserve	NumFiles	NumRWFiles
25	15M	95M	1	1
7	10M	10M	1	1
1	10N	20M	1	1
1	10M	40M	1	1

Stripingon	StripeSize	BlkTypes	OK ToDrop
T	1K	1H,1248F,32D,177A, 128M	N
T	1K	1H,64F,16A	N
T	1K	1H	Y
T	1K	1H	Y

You can decrease the dbspace size only if the truncated portion is not in use. Use **sp_iqdbspaceinfo** to determine which blocks are in use by the objects on a dbspace.

Adding Database Objects

Create two tables in the `dbspacedb` database, create indexes, and add some data:

```
CREATE TABLE t1(c1 int);
CREATE TABLE t2(c1 int);
CREATE hg INDEX t1c1hg ON t1(c1);
CREATE hng INDEX t2c1hng ON t2(c1);
INSERT t1 VALUES(1);
INSERT t2 VALUES(2);
COMMIT;
```

Displaying Information about Dbspaces

Display information about all dbspaces in the `dbspacedb` database. The following example shows dbspaces in the iqdemo database. Output is divided into two parts to improve readability:

```
sp_iqdbspace;
```

DBSpaceName	DBSpace-Type	Writable	Online	Usage	To-tal-Size	Reserve
IQ_SYSTEM_MAIN	MAIN	T	T	25	10M	100M
IQ_SYSTEM_TEMP	TEMPORARY	T	T	7	10M	10M
dbspacedb2	MAIN	T	T	1	10N	20M
dbspacedb3	MAIN	T	T	1	10M	40M

NumFiles	NumRWFiles	Stripingon	StripeSize	BlkTypes	OK To-Drop
1	1	T	1K	1H,1248F,32D, 177A,128M	N
1	1	T	1K	1H,64F,16A	N
1	1	T	1K	1H	Y
1	1	T	1K	1H	Y

Display information about object placement and space usage for a specific dbspace.

Note: The next two examples show objects in the `iqdemo` database to better illustrate output. Note that `iqdemo` includes a sample user dbspace named `iq_main` that may not be present in your own databases.

The following output is divided into parts to improve readability:

```
sp_iqdbspaceinfo;
```

dbspace_name	object_type	owner	object_name	object_id	id
iq_main	table	DBA	emp1	3,813	743
iq_main	table	DBA	iq_dummy	3,801	742

dbspace_name	object_type	owner	object_name	object_id	id
iq_main	table	DBA	sale	3,822	744
iq_main	table	GROUPO	Contacts	3,662	734
iq_main	table	GROUPO	Customers	3,639	733
iq_main	table	GROUPO	Departments	3,756	740
iq_main	table	GROUPO	Employees	3,765	741
iq_main	table	GROUPO	FinancialCodes	3,736	738
iq_main	table	GROUPO	FinancialData	3,745	739
iq_main	table	GROUPO	Products	3,717	737
iq_main	table	GROUPO	SalesOrderItems	3,704	736
iq_main	table	GROUPO	SalesOrders	3,689	735

columns	indexes	metadata	primary_key
96K	0B	1.37M	0B
24K	0B	464K	0B
96K	0B	1.22M	0B
288K	0B	5.45M	24K
240K	48K	4.63M	24K
72K	0B	1.78M	24K
408K	0B	8.03M	24K
72K	0B	1.53M	24K
96K	0B	2.19M	24K
272K	192K	4.67M	24K
120K	0B	2.7M	24K
144K	0B	3.35M	24K

unique_ constraint	foreign_key	dbspace_online
0B	0B	Y
0B	0B	Y
0B	0B	Y
0B	48K	Y
0B	0B	Y
0B	48K	Y
0B	48K	Y
0B	0B	Y
0B	48K	Y
0B	0B	Y
0B	104K	Y
0B	144K	Y

Use the **sp_iqindexinfo** system stored procedure to display object placement and space usage for a specific table or index. The following information is from the `iqdemo` database.

```
sp_iqindexinfo 'table GROUPO.Customers';
```

Object	DBSpace-Name	ObjSize	DBSpPct	MinBlk	MaxBlk
GROUPO.Customers	iq_main	200K	1	1,045,460	1,051,032
GROUPO.Customers.ASIQ_IDX_T733_C10_FP	iq_main	440K	1	1,046,689	1,047,147
GROUPO.Customers.ASIQ_IDX_T733_C1_FP	iq_main	440K	1	1,046,641	1,047,213
GROUPO.Customers.ASIQ_IDX_T733_C2_FP	iq_main	440K	1	1,046,961	1,047,203

Object	DBSpace-Name	ObjSize	DBSpPct	MinBlk	MaxBlk
GROUPO.Customers.ASIQ_IDX_T733_C3_FP	iq_main	440K	1	1,046,833	1,047,196
GROUPO.Customers.ASIQ_IDX_T733_C4_FP	iq_main	440K	1	1,046,737	1,047,189
GROUPO.Customers.ASIQ_IDX_T733_C5_FP	iq_main	440K	1	1,046,929	1,047,182
GROUPO.Customers.ASIQ_IDX_T733_C6_FP	iq_main	440K	1	1,047,009	1,047,175
GROUPO.Customers.ASIQ_IDX_T733_C7_FP	iq_main	440K	1	1,046,945	1,047,168
GROUPO.Customers.ASIQ_IDX_T733_C8_FP	iq_main	440K	1	1,046,785	1,047,161
GROUPO.Customers.ASIQ_IDX_T733_C9_FP	iq_main	440K	1	1,046,881	1,047,154
GROUPO.Customers.ASIQ_IDX_T733_I11_HG	iq_main	152K	1	1,047,121	1,047,206
GROUPO.Customers.IX_customer_name	iq_main	304K	1	1,050,995	1,051,038

For the full syntax of the **sp_iqdbspace**, **sp_iqdbspaceinfo**, and **sp_iqindexinfo** system stored procedures, see *Reference: Building Blocks, Tables, and Procedures*.

Data Storage

A DBA can determine which tables and indexes reside on a given dbspace, relocate objects to other dbspaces, and drop any dbspace after emptying it of data.

A DBA can also define the number of writes to each dbspace before the disk striping algorithm moves to the next stripe.

Dbfile Attributes and Operations

A dbfile has read-write or read-only status.

A dbfile is read-write when it is added, and its runtime read-write status depends on both the read-write status of the dbspace and of the dbfile. The administrator can alter the read-write/read-only status of a dbfile, but cannot alter the online/offline status of a dbfile.

Operations that can be performed on dbfiles include adding, dropping, renaming logical name, and renaming the file path.

See also

- *ALTER DBSPACE Statement* on page 416

Adding a File to the RLV Dbspace

You may wish to add a file to the RLV dbspace for extra capacity in storing RLV transaction logs.

Prerequisites

- SAP Sybase IQ server has a simplex database.
- A single RLV dbspace exists on the database, and is online.
- If the dbspace is read-write, no RLV-enabled objects exist.

Task

Use the statement **ALTER DBSPACE** *<dbspace name>* **ADD FILE** *<filename>*

```
ALTER DBSPACE d1 ADD FILE 'rlv2.iq'
```

Because of the nature of in-memory RLV storage, you cannot specify files as being READONLY.

Dropping a File from the RLV Dbspace

You can remove a file from the RLV dbspace, provided that it is not the only file, and it is not in use.

Prerequisites

- SAP Sybase IQ server has a simplex database.
- A single RLV dbspace exists on the database.
- The RLV dbspace is read-only, or the file is not in use if the dbspace is read-write.

Task

Use the statement **ALTER DBSPACE** *<dbspace name>* **DROP FILE** *<dbspace filename>*

```
ALTER DBSPACE d1 DROP FILE rlv2
```

Dropping Dbfiles from IQ_SYSTEM_TEMP

Issue a **DROP DBSPACE** command to remove a dbfile.

Prerequisites

To drop a dbfile:

- Requires the MANAGE ANY DBSPACE system privilege.
- The dbfile cannot contain any data from user tables. SAP Sybase IQ does not allow you to drop a dbfile unless it is empty.
- There must be at least one read-write dbfile available in the dbspace.
- You must commit or roll back all transactions that use older versions of tables on the dbfile.

Task

1. Search the global temporary table schema using the system tables. Run DROP TABLE commands to relocate all objects resident on the dbfile.
   ```
   SELECT * FROM systable WHERE table_type = 'GBL TEMP'
   ```
2. Verify that all space on the dbfile is free.
   ```
   sp_iqfile IQ_SYSTEM_TEMP
   ```
   If the `OkToDrop` column contains "Y", you may drop the dbfile.
3. Drop the dbfile. For a multiplex database, use an absolute path.
   ```
   ALTER DBSPACE IQ_SYSTEM_TEMP DROP FILE FileHist3
   ```

Dbfiles and Output Files

The `SYSDBFILE` view shows all the dbfiles in your database, including the catalog dbspace file, the IQ message file, dbfiles in the IQ main and temporary dbspaces, the transaction log file, and the SA temporary file.

Files that are not dbfiles do not appear in the `SYSDBFILE` view. These include files that may be generated on server startup, such as the console log (specified by the **-o** switch) and the SQL log (specified by **-zo**). These log files do appear as database properties or server properties and may be examined by stored procedures such as **sa_db_properties**( ) or the system function **db_property**( ). (For syntax, see *Table 14-1* on page 628.)

Dbspace Attributes and Operations

Dbspace statuses can be online, offline, or dynamically offline.

Dynamically offline means that the dbspace is marked offline in memory, as opposed to marked offline in the catalog. If a database starts and one or more dbfiles cannot be opened, the database starts but the dbspace is marked dynamically offline. An administrator can use

ALTER DBSPACE ONLINE to bring a dbspace back online after fixing a problem, but this changes only the dbspace's in-memory status.

Note: Table data is inaccessible if any indexes, data, or partitions exist in an offline dbspace.

In addition to online, offline, or dynamically offline status, a dbspace also has read-write or read-only status. When created, a dbspace is online and read-write.

A dbspace also has striping attributes. An administrator may specify whether striping is on, and the stripe size.

You can create, alter, or drop dbspaces.

For multiplex dbspaces, see *Administration: Multiplex*. To change the status of a dbspace in Sybase Control Center, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

Dbspace Renaming Guidelines

You can rename a dbspace or dbfile name, but you cannot rename or drop catalog dbspace SYSTEM, IQ main dbspace IQ_SYSTEM_MAIN, IQ temporary dbspace IQ_SYSTEM_TEMP, shared temporary dbspace IQ_SHARED_TEMP, and IQ message dbspace IQ_SYSTEM_MSG.

You can rename the logical name of files in IQ_SYSTEM_MAIN, IQ_SYSTEM_TEMP, and you can change the logical name of IQ_SYSTEM_MSG files, but you cannot change the logical name of files in SYSTEM. You cannot use ALTER DBSPACE RENAME TO to rename dbspaces IQ_SYSTEM_MAIN or IQ_SYSTEM_TEMP, IQ_SYSTEM_MSG, or SYSTEM.

Additional Dbspaces

Create a dbspace using the **CREATE DBSPACE** statement.

A new dbspace can be on the same or a different disk drive as the existing dbspaces. Requires the MANAGE ANY DBSPACE system privilege.

See *Reference: Building Blocks, Tables, and Procedures > Physical Limitations* for the maximum sizes of dbspaces on raw devices and operating system files. On some platforms, you must enable large file system files to reach the maximum size.

You can specify **SIZE** and **RESERVE** only for the IQ store and IQ temporary store, not for the catalog store.

Create main stores on raw devices.

When you specify a raw device for a new dbspace, SAP Sybase IQ automatically determines its file size and allocates the entire device for use as an IQ store. This may have unpredictable results on a file device.

If you indicate that the device is not raw, you can then specify the file size. The wizard verifies that the given path exists. If the path doesn't exist, SAP Sybase IQ returns an error.

How the Number of Dbspaces Affects Resource Use and Performance

The maximum number of dbspaces per database is an operating system limit that you can adjust; the maximum is 2,047 dbspaces per IQ database, plus a maximum of 12 dbspaces for the catalog store. Increasing the number of dbspaces has no real impact on memory use or performance; avoid situations where you approach the maximum.

Note: On HP and AIX platforms, overlapped I/O performance improves when you divide data among more dbspaces.

When data is stored on raw partitions, you can have one dbspace per drive. See *Reference: Building Blocks, Tables, and Procedures > Physical Limitations* for dbspace size limits.

When data is stored in a file system, you can take advantage of striping in the storage system. If you use operating system or hardware striping on a multiuser system, your stripe size should be a minimum of 1MB, or the highest size possible. The stripe size should be several times your IQ page size. You can also configure IQ to perform software striping.

Before adding dbspaces, you may want to estimate your space requirements. For the most efficient resource use, make your dbspaces small enough to fit on your backup media, and large enough to fill up the disk.

Example

The following command creates a new dbspace called `library` which points to an operating system file named `library.iq` in the same directory as the `IQ_SYSTEM_MAIN` dbspace:

```
CREATE DBSPACE library
USING FILE library
'library.iq' SIZE 100 MB IQ STORE
```

To create a dbspace in Sybase Control Center, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

After you add or drop a dbspace, issue a **CHECKPOINT**. System recovery begins after the most recent checkpoint.

Guidelines for Dropping a Dbspace

Issue a **DROP DBSPACE** command to remove a database file.

To drop a dbspace:

- Requires the MANAGE ANY DBSPACE system privilege.
- It cannot contain any data from user tables. SAP Sybase IQ does not allow you to drop a dbspace unless it is empty.
- It cannot be a required dbspace: `SYSTEM`, `IQ_SYSTEM_MAIN`, `IQ_SYSTEM_TEMP`, or `IQ_SYSTEM_MSG`. These dbspaces can never be dropped, but you may drop other

dbspaces from the IQ main store or catalog store if the dbspace contains no user-created objects.

To empty a dbspace, you must:

- Relocate or drop all objects resident on the dbspace.
- Commit or roll back only transactions that are using older versions of tables.

Because of the way SAP Sybase IQ fills dbspaces with data, it is unlikely that a dbspace will become empty only after explicitly relocating tables, especially if disk striping is in use. Typically, you cannot empty a dbspace by truncating the tables in it, as even an empty table takes some space. To relocate the tables, use **ALTER TABLE MOVE**.

If you relocate a table while other users are reading from it, the normal rules of table versioning apply, that is, old table versions persist until the transactions of the readers complete.

A DBA can determine the dbspace in which tables and indexes are located by running the stored procedures **sp_iqspaceinfo**, **sp_iqdbspaceinfo**, and **sp_iqindexinfo**. These procedures show the number of blocks used by each table and index in each dbspace.

To find out whether you can drop a particular dbspace, run **sp_iqdbspace**. Look at the Block Types column (Blk Types), which tells you the contents of each dbspace. You can drop a dbspace if it contains block types “H,” “F,” “R,” “B,” “X,” and “C,” but not other block types.

Block type “A” is data from active table versions. Use **sp_iqdbspaceinfo** to determine which tables to relocate.

Block type “O” indicates old versions that may still be in use. You must roll back or commit active connections to release this space. Block type “M” indicates multiplex.

For instructions on using Sybase Control Center to delete a dbspace, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

Read-only and Read-write Dbspaces and Files

A file is read-only when either the file status is read-only or the file status is read-write, but the owning dbspace status is read-only.

Altering a dbspace to read-only does not alter the catalog status of its associated files to read-only, but does make the associated files read-only at the operating system level. In other words, the file’s catalog read-only or read-write status remains the same, but data in the file cannot be modified.

For a read-only dbspace, the administrator can:

- Add a file
- Rename the file path of a dbfile in the dbspace (requires main dbspaces are offline)
- Drop an empty file
- Rename the dbspace or dbfile in the dbspace

Status of Dbspaces and Associated Files

A dbspace and its associated files can have individual read-only (RO) or read-write (RW) status, for example:

Object	Status	Effective status	Dbspace of Table	Status of Table
dbspace1	RW	RW	dbspace1	RW
- file1	RO	RO		
- file2	RW	RW		
dbspace2	RO	RO	dbspace2	RO
- file1	RO	RO		
- file2	RW	RO		
dbspace3	RW	RO	dbspace3	RO
- file1	RO	RO		
- file2	RO	RO		

A table is read-only when it is assigned to a read-only dbspace. A table partition is read-only when the partition is assigned to a read-only dbspace. No data modifications such as insert, delete, update, load, truncate table, and insert/delete/update through an updatable cursor are allowed to a read-only table or read-only table partition. No DDL operations such as ALTER TABLE add/drop column, create/drop index are allowed on a read-only table or read-only table partition.

Attempts to write to a read-only dbspace are detected when the modified pages are flushed to disk. Pages modified during an **INSERT...VALUES** statement are not written to the database until the next command that is not an **INSERT...VALUES** statement. (**INSERT...VALUES** is the only command that behaves this way.) SAP Sybase IQ returns an error for DDL operations on a read-only table or read-only table partition.

Allowed Dbspace Transformations

Your ability to change a dbspace configuration depends on the type of alter operation and certain attributes of the dbspace. Alter operations are governed by the state of the dbspace (online or offline), the read-write status of the dbspace, and the type of dbspace.

Table 7. Allowed Dbspace Configuration Transformations

State	Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
Online dbspace			
	ALTER DBSPACE OFFLINE	Yes, if RO	No
	ALTER DBSPACE ONLINE	No	Yes for IQ_SYSTEM_MAIN, No for IQ_SHARED_TEMP and IQ_SYSTEM_TEMP
	ALTER DBSPACE READONLY	Yes, if it is RW	No
	ALTER DBSPACE READWRITE	Yes, if it is RO	No
	ALTER STRIPING or STRIPESIZEKB	Yes	Yes on simplex and on multiplex coordinator
	RENAME DBSPACE	Yes	No
	ADD FILE	Yes	Yes

State	Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
	DROP FILE	Yes, if empty and RO	Yes, if empty and RO. Dropping files in IQ_SHARED_TEMP requires that this operation be done only on the coordinator once it is started in a single-node mode. Dropping files in IQ_SHARED_TEMP in simplex is also supported. The first file that is made RW in IQ_SHARED_TEMP must be the last file to be dropped.
	ALTER FILE READONLY	Yes, if RW	Yes, if RW and not the last RW dbfile. Making files in IQ_SHARED_TEMP read-only is not allowed.
	ALTER FILE READWRITE	Yes, if RO	Yes, if RO
	ALTER FILE SIZE	Yes, if RW dbspace and dbfile	Yes, if RW
	ALTER FILE RENAME LOGICAL NAME	Yes	Yes
	ALTER FILE RENAME PATH	No	Yes for IQ_SHARED_TEMP and IQ_SYSTEM_TEMP; takes effect when database restarts. No for IQ_SYSTEM_MAIN.
Offline dbspace			
	ALTER DBSPACE OFFLINE	No	N/A
	ALTER DBSPACE ONLINE	Yes	N/A

State	Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
	ALTER DBSPACE READONLY	No	N/A
	ALTER DBSPACE READWRITE	No	N/A
	ALTER STRIPING or STRIPESIZEKB	Yes	N/A
	RENAME DBSPACE	Yes	N/A
	ADD FILE	No	N/A
	DROP FILE	Yes, if empty	N/A
	ALTER FILE RO	Yes	N/A
	ALTER FILE RW	Yes	N/A
	ALTER FILE SIZE	No	N/A
	ALTER FILE RENAME LOGICAL NAME	Yes	N/A
	ALTER FILE RENAME PATH	Yes	N/A
Dynamically offline dbspace			
	ALTER DBSPACE OFFLINE	Yes, if RO	N/A
	ALTER DBSPACE ONLINE	Yes	N/A for IQ_SYSTEM_MAIN and IQ_SYSTEM_TEMP, Yes for IQ_SHARED_TEMP
	ALTER DBSPACE READONLY	Yes, if RW	N/A
	ALTER DBSPACE READWRITE	No	N/A
	ALTER STRIPING or STRIPESIZEKB	Yes	N/A

State	Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
	RENAME DBSPACE	Yes	N/A
	ADD FILE	No	N/A
	DROP FILE	Yes, if empty	N/A
	ALTER FILE READONLY	No	N/A
	ALTER FILE READWRITE	No	N/A
	ALTER FILE SIZE	No	N/A
	ALTER FILE RENAME LOGICAL NAME	Yes	N/A
	ALTER FILE RENAME PATH	No	N/A
Read-only dbspace			
	ALTER DBSPACE OFFLINE	Yes, if online	N/A
	ALTER DBSPACE ONLINE	Yes, if offline	N/A
	ALTER DBSPACE READONLY	No	N/A
	ALTER DBSPACE READWRITE	Yes, if online	N/A
	ALTER STRIPING and STRIPESIZEKB	Yes	N/A
	RENAME DBSPACE	Yes	N/A
	ADD FILE	Yes	N/A
	DROP FILE	Yes, if empty	N/A
	ALTER FILE READONLY	Yes, if RW	N/A
	ALTER FILE READWRITE	Yes, if RO	N/A
	ALTER FILE SIZE	No	N/A
	ALTER FILE RENAME LOGICAL NAME	Yes	N/A

State	Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
	ALTER FILE RENAME PATH	Yes, if offline	N/A
Read-write dbspace			
	ALTER DBSPACE OFFLINE	No	No
	ALTER DBSPACE ONLINE	Yes, if dynamically offline	No for IQ_SYSTEM_MAIN and IQ_SYSTEM_TEMP, Yes if dynamically offline for IQ_SHARED_TEMP
	ALTER DBSPACE READONLY	Yes	No
	ALTER DBSPACE READWRITE	No	No
	ALTER STRIPING and STRIPESIZEKB	Yes	Yes
	RENAME DBSPACE	Yes	No
	ADD FILE	Yes	Yes
	DROP FILE	Yes, if empty	Yes, if empty and RO. Dropping files in IQ_SHARED_TEMP requires that this operation be done only on the coordinator once it is started in a single-node mode. Dropping files in IQ_SHARED_TEMP in simplex is also supported. The first file that is made RW in IQ_SHARED_TEMP must be the last file dropped.

State	Alter Type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
	ALTER FILE READONLY	Yes, if RW	Yes, if RW. Making files in IQ_SHARED_TEMP readonly is not allowed.
	ALTER FILE READWRITE	Yes, if RO	Yes, if RO
	ALTER FILE SIZE	Yes, if RW	Yes, if RW
	ALTER FILE RENAME LOGICAL NAME	Yes	Yes
	ALTER FILE RENAME PATH	No	Yes, takes effect when database restarts
Read-only file			
	ALTER FILE READONLY	No	No
	ALTER FILE READWRITE	Yes	Yes
	ALTER FILE SIZE	No	No
	ALTER FILE RENAME LOGICAL NAME	Yes	Yes
	ALTER FILE RENAME PATH	Yes, if offline	Yes, takes effect when database restarts
Read-write file			
	ALTER FILE READONLY	Yes	Yes
	ALTER FILE READWRITE	No	No
	ALTER FILE SIZE	Yes, if dbspace is RW and ONLINE	Yes
	ALTER FILE RENAME LOGICAL NAME	Yes	Yes
	ALTER FILE RENAME PATH	No	Yes, takes effect when database restarts

Note:

- Dynamically offline means the dbspace is marked offline in memory, as opposed to marked offline in the catalog.
- A read-only (RO) IQ_SYSTEM_MAIN dbfile can be dynamically offline.
- For IQ_SYSTEM_MSG, the only modification that is permitted is to rename the path, which is done using the command `ALTER DBSPACE IQ_SYSTEM_MSG RENAME 'filepath'`.

Manage SAP Sybase IQ Database Objects

SAP Sybase IQ lets you create, alter, and delete database objects such as tables and views.

Note: SAP Sybase IQ consists of a catalog store and an IQ store. This section explains how you create objects in your IQ store. Tables created in the catalog store have the characteristics of SQL Anywhere tables. To create tables in the catalog store, see the SQL Anywhere documentation.

SAP Sybase IQ Database Design

The right database design enhances data usability and the speed of data retrieval.

Before you create the database, you must decide where to store the data, how much space your database requires, and who will be able to define or modify database objects. SAP Sybase IQ provides tools and processes to help.

Sybase PowerDesigner® can help you design your database, by building a conceptual, physical, or object-oriented data model, and then generating the database from the model. It also lets you reverse engineer, creating a model from an existing database.

No matter which design tool is used, the database administrator (DBA) generally designs the database and defines its contents. To create an effective design, the DBA needs to work with individuals throughout your organization to understand how data will be used. The DBA also needs to understand the concepts underlying IQ databases.

A SAP Sybase IQ database is a relational database that is optimized for use as a data warehouse. As a relational database, it consists of a set of related tables that organize the data; as a data warehouse, it provides efficient access to very large sets of data by means of indexes.

When you create a database, you specify the structure of these tables, the types of data allowed in them, the relationships among tables, the indexes that store the table data, and views that control who has access to the data. Before creating an IQ database, be sure you understand the relational database and data warehousing concepts described in *Introduction to SAP Sybase IQ*.

View Management

Views are computed tables.

You can use views to show database users exactly the information you want to present, in a format you can control.

Similarities between Views and Base Tables

Views are similar to the permanent tables of the database (a permanent table is also called a *base table*) in many ways:

- You can assign access permissions to views just as to base tables.
- You can perform **SELECT** queries on views.
- You can perform **INSERT** and **DELETE** operations on some views.
- You can create views based on other views.

Differences Between Views and Permanent Tables

There are some differences between views and permanent tables:

- You cannot create indexes on views.
- **INSERT**, **DELETE**, and **UPDATE** operations can only be performed on certain views.
- You cannot assign integrity constraints and keys to views.
- Views refer to the information in base tables, but do not hold copies of that information. Views are recomputed each time you invoke them.

Benefits of Tailoring Data Access

Views are used to tailor access to data in the database. Tailoring access serves several purposes:

- By not allowing access to information that is not relevant.
- By presenting users and application developers with data in a more easily understood form than in the base tables.
- By centralizing in the database the definition of common queries.

Creating Views

A **SELECT** statement operates on one or more tables and produces a result set that is also a table.

Like a base table, a result set from a **SELECT** query has columns and rows. A view gives a name to a particular query, and holds the definition in the database system tables.

Example

Suppose that you frequently need to list the number of employees in each department. You can get this list with the following statement:

```
SELECT DepartmentID, COUNT(*)
FROM Employees
GROUP BY DepartmentID
```

You can create a view containing the results of this statement as follows:

```
CREATE VIEW DepartmentSize AS
SELECT DepartmentID, COUNT(*)
```

```
FROM Employees
GROUP BY DepartmentID
```

The information in a view is not stored separately in the database. Each time you refer to the view, the associated **SELECT** statement is executed to retrieve the appropriate data.

On one hand, this is good because it means that if someone modifies the `Employees` table, the information in the `DepartmentSize` view will be automatically up to date. On the other hand, complicated **SELECT** statements may increase the amount of time SQL requires to find the correct information every time you use the view.

Guidelines for Using Views

There are certain restrictions, both on the **SELECT** statements you use to create views, and on your ability to insert into, delete from, or update them.

Restrictions on SELECT Statements

You cannot use an **ORDER BY** clause in the **SELECT** query. A characteristic of relational tables is that there is no significance to the ordering of the rows or columns, and using an **ORDER BY** clause imposes an order on the rows of the view. You can use the **GROUP BY** clause, subqueries, and joins in view definitions.

Scalar value subqueries are supported only within the top-level **SELECT** list (not in a view, a derived table, or a subquery). Sometimes views or derived tables used in the **FROM** clause of the top level **SELECT** are simple enough that they can be "flattened" up into the top level **SELECT**. As a result of this, the preceding rule is actually enforced only for subqueries, nonflattened views, and nonflattened derived tables. For example:

```
CREATE VIEW test_view AS SELECT testkey,(SELECT COUNT(*) FROM
tagtests WHERE tagtests.testkey = testtrd.testkey ) FROM
testtrd
```

```
SELECT * FROM test_view
Msg 21, Level 14, State 0:
SQL Anywhere Error -1005004: Subqueries are allowed only as arguments
of
comparisons, IN, and EXISTS,
-- (opt_Select.cxx 2101)
```

To develop a view, tune the **SELECT** query by itself until it provides exactly the results you need in the format you want. Once you have the correct **SELECT** query, you can add a phrase in front of the query to create the view. For example:

```
CREATE VIEW viewname AS
```

Guidelines for Inserting and Deleting from Views

UPDATE, **INSERT**, and **DELETE** statements are allowed on some views, but not on others, depending on their associated **SELECT** statement.

You cannot update, insert into or delete from views that contain:

- Aggregate functions, such as **COUNT(*)**
- A **GROUP BY** clause in the **SELECT** statement
- A **UNION** operation

In all these cases, there is no way to translate the **UPDATE**, **INSERT**, or **DELETE** into an action on the underlying tables.

Warning! Do not delete views owned by the dbo user ID, which owns system objects. Deleting such views or changing them into tables may cause unexpected problems.

Guidelines for Modifying Views

You can modify a view using the **ALTER VIEW** statement.

The **ALTER VIEW** statement replaces a view definition with a new definition; it does not modify an existing view definition.

The **ALTER VIEW** statement maintains the permissions on the view.

Example

For example, to replace the column names with more informative names in the `DepartmentSize` view described above, you could use the following statement:

```
ALTER VIEW DepartmentSize
   (DepartmentID, NumEmployees)
AS
  SELECT DepartmentID, COUNT(*)
  FROM Employees
  GROUP BY DepartmentID
```

Permissions on Views

A user may perform an operation through a view if one or more of the following are true:

- The appropriate permissions on the view for the operation have been granted to the user by a DBA.
- The user has the appropriate permissions on all the objects (such as base tables, views, procedures) for the operation.
- The user was granted appropriate permissions for the operation on the view by a non-DBA user. This user must be either the owner of the view or have WITH GRANT OPTION of the appropriate permission(s) on the view. The owner of the view must be either:
  - a DBA, or
  - a non-DBA, but also the owner of all the objects referred to by the view, or
  - a non-DBA, and not the owner of some or all of the objects referred to by the view, but the view owner has SELECT permission WITH GRANT OPTION on the objects not owned and any other required permission(s) WITH GRANT OPTION on the objects not owned for the operation.

Instead of the owner having permission(s) WITH GRANT OPTION on the objects, permission(s) may have been granted to PUBLIC. This includes SELECT permission on system tables.

UPDATE permissions can be granted only on an entire view. Unlike tables, UPDATE permissions cannot be granted on individual columns within a view.

How to Delete Views

To delete a view from the database in Interactive SQL , use the **DROP** statement. The following statement removes the DepartmentSize view:

```
DROP VIEW DepartmentSize
```

View Information in System Views

Information about views in a database is in the system view SYS.SYSVIEW.

See *Reference: Building Blocks, Tables, and Procedures > System Tables and Views*.

You can use Interactive SQL to browse the information in this view. Type the following statement in the **dbisql** command window to see all the columns in the SYS.SYSVIEW view:

```
SELECT *
FROM SYS.SYSVIEW
```

To extract a text file containing the definition of a specific view, use a statement such as the following:

```
SELECT view_def FROM SYS.SYSVIEW
WHERE view_object_id = 1583;
OUTPUT TO viewtext.sql
FORMAT ASCII
```

Table Management

When you create a database, the only tables in it are the *system tables*, which hold the database schema.

You may need to create, alter, and delete tables from a database. You can execute examples in documentation using **dbisql**, but the SQL statements are independent of the administration tool you are using.

You can create command files containing the **CREATE TABLE** and **ALTER TABLE** statements that define the tables in your database and store them in a source code control system. The command files allow you to re-create the database when necessary. They also let you create tables in a standardized way, which you can copy and revise.

Guidelines for Creating Tables

Create tables using Interactive SQL.

Table Creation with Interactive SQL

The SQL statement for creating tables is **CREATE TABLE**.

This section describes how to use the **CREATE TABLE** statement. The examples in this section use the sample database. To try the examples, run **dbisql** and connect to the demo database with user ID `DBA` and password `sql`.

You can create tables with other tools in addition to Interactive SQL. The SQL statements described here are independent of the tool you are using.

Example

The following statement creates a new, permanent IQ table to describe qualifications of employees within a company. The table has columns to hold an identifying number, a name, and a type (say `technical` or `administrative`) for each skill.

```
CREATE TABLE skill (
skill_id INTEGER NOT NULL,
skill_name CHAR( 20 ) NOT NULL,
skill_type CHAR( 20 ) NOT NULL
)
```

You can execute this command by typing it into the **dbisql** command window, and pressing the execute key (F9).

- Each column has a *data type*. The `skill_id` is an `integer` (like 101), the `skill_name` is a fixed-width `CHARACTER` string containing up to 20 characters, and so on.
- The phrase NOT NULL after their data types indicates that all columns in this example must contain a value.
- In general, you would not create a table that has no primary key.

By internally executing the **COMMIT** statement before creating the table, SAP Sybase IQ makes permanent all previous changes to the database. There is also a **COMMIT** after the table is created.

Warning! Altering or creating global or base tables can interfere with other users of the database. For large tables, **ALTER** or **CREATE TABLE** can be a time-consuming operation. **CREATE TABLE** processing delays execution of other IQ processes until the statement completes. Although you can execute **ALTER TABLE** statements while other connections are active, you cannot execute them while any other connection uses the table to be altered. **ALTER TABLE** processing excludes other requests referencing the table being offered while the statement processes.

Specifying Data Types

When you create a table, you specify the type of data that each column holds.

You can also define customized data types for your database. See *Reference: Building Blocks, Tables, and Procedures* for a list of supported data types. See the **CREATE DOMAIN** statement in *Reference: Statements and Options* for details on how to create a customized data type.

Types of Tables

SAP Sybase IQ recognizes four types of tables.

- Base tables
- Local temporary tables
- Global temporary tables
- Join virtual tables

Base Tables are Permanent

Base tables are sometimes called main, persistent, or permanent tables because they are a permanent part of the database until you drop them explicitly.

They remain in the database over user disconnects, server restart, and recovery. Base tables and the data in them are accessible to all users who have the appropriate permissions. The **CREATE TABLE** statement shown in the previous example creates a base table.

Creating Temporary Tables

There are two types of temporary tables, global and local.

You *create* a global temporary table, using the **GLOBAL TEMPORARY** option of **CREATE TABLE**.

When you create a global temporary table, it exists in the database until it is explicitly removed by a **DROP TABLE** statement.

A database contains only one definition of a global temporary table, just as it does for a base table. However, each user has a separate instance of the data in a global temporary table. Those rows are visible only to the connection that inserts them. They are deleted when the connection ends, or commits. A given connection inherits the schema of a global temporary table as it exists when the user first refers to the table. Global temporary tables created on a multiplex server are also created on all other multiplex servers.

To select into a temporary table, use syntax like the following:

```
SELECT * INTO #TableTemp FROM lineitem
WHERE l_discount < 0.5
```

Note: SAP Sybase strongly recommends that, when writing scripts that **SELECT INTO** a temporary table, you wrap any select list item that is not a base column in a **CAST** expression. This guarantees that the temporary table's column data type is the data type desired.

You *declare* a local temporary table for your connection only, using the **DECLARE LOCAL TEMPORARY TABLE** statement. A local temporary table exists until the connection ends or commits, or within a compound statement in which it is declared. The table and its data are completely inaccessible to other users.

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 the following sequence:

```
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.

Dropping and Altering Global Temporary Tables

You drop a global temporary table just as you would a base table, with the **DROP TABLE** statement.

You cannot drop or alter a global temporary table while other connections are using the table.

Placement of Tables

SAP Sybase IQ creates tables in your current database.

If you are connected to an SAP Sybase IQ database, tables are placed as follows:

Table 8. Table Placement

Type of Table	Permitted Placement	Default Placement
Permanent	IQ store, catalog store	IQ store
Global temporary	IQ temporary store, catalog store	IQ temporary store
Local temporary	IQ temporary store or catalog store; only visible to user who creates it	IQ temporary store

Automatic Index Creation for IQ Tables

You can automate indexing for certain columns by creating a table with either **PRIMARY KEY** or **UNIQUE** constraints.

These options cause SAP Sybase IQ to create an **HG** index for the column that enforces uniqueness.

If you use the **ALTER TABLE** command to add a **UNIQUE** column to an existing table, or to designate an existing column as **UNIQUE**, an **HG** index is created automatically.

Optimizing Storage and Query Performance

When you create a permanent table in an SAP Sybase IQ database, SAP Sybase IQ automatically stores permanent table columns in default indexes that facilitates a type of query called a projection.

SAP Sybase IQ optimizes this structure for query performance and storage requirements, based on these factors:

- The **IQ UNIQUE** option (**CREATE TABLE** or plug-in Column Properties page)
- The data type of the column and its width
- The **IQ PAGE SIZE** option (**CREATE DATABASE** or Sybase Control Center Create Databases wizard)

Effect of IQ UNIQUE

`IQ UNIQUE` is an optional column constraint that explicitly defines the cardinality of a column and determines whether the column loads as `Flat FP` or `NBIT`.

IQ UNIQUE defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit. 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 and FP_NBIT_LOOKUP_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.

Indexes and IQ UNIQUE

If you estimate **IQ UNIQUE** incorrectly, there is no penalty for loads; the Optimizer simply uses the next larger index.

For queries, if you estimate **IQ UNIQUE** incorrectly and you have an **HG**, **LF**, or storage-optimized default index, the Optimizer ignores the **IQ UNIQUE** value and uses the actual number of values in the index. If you do not have one of these indexes and your estimate is wrong by a significant amount (for example, if you specify `IQ UNIQUE 1000000` when the actual number of unique values is 12 million), query performance may suffer.

To change the value of **IQ UNIQUE** for an existing index, run the **sp_iqrebuildindex** procedure.

Difference Between UNIQUE and IQ UNIQUE

I`IQ UNIQUE(`*`n`*`)` approximates the number of distinct values in a given column. An `IQ UNIQUE(`*`n`*`)` value explicitly set to '0' loads the column as `Flat FP`. A column without an `IQ UNIQUE` or a column with an `IQ UNIQUE(`*`n`*`)` value less than or equal to the limits defined by the `FP_NBIT_AUTOSIZE_LIMIT` and `FP_NBIT_LOOKUP_MB` options implicitly loads as `NBIT`. Each distinct value can appear many times. For example, in the `employee` table, a limited set of distinct values could appear in the `state` column, but each of those values could appear in many rows.

By contrast, when you specify **UNIQUE** or **PRIMARY KEY**, each value can occur only once in that column. For example, in the `employee` table, each value of `ss_number`, the employee's social security number, can occur just once throughout that column. This uniqueness extends to NULL values. Thus, a column specified as **UNIQUE** must also have the constraint **NOT NULL**.

Guidelines for Altering Tables

This section describes how to change the structure of a table using the **ALTER TABLE** statement.

Example 1

The following command adds a column to the `skill` table to allow space for an optional description of the skill:

```
ALTER TABLE skill
ADD skill_description CHAR( 254 )
```

Example 2

The following statement changes the name of the `skill_type` column to `classification`:

```
ALTER TABLE skill
RENAME skill_type TO classification
```

Example 3
The following statement deletes the `classification` column.

```
ALTER TABLE skill
DELETE classification
```

Example 4
The following statement changes the name of the entire table:

```
ALTER TABLE skill
RENAME qualification
```

These examples show how to change the structure of the database. The **ALTER TABLE** statement can change many characteristics of a table—foreign keys can be added or deleted, and so on. However, you cannot use **MODIFY** to change table or column constraints. Instead, you must **DELETE** the old constraint and **ADD** the new one. In all these cases, once you make the change, stored procedures, views, and any other item referring to this column will no longer work.

Guidelines for Dropping Tables

Dropping a table deletes all the records in the table and then removes the table definition.

The following **DROP TABLE** statement deletes all the records in the `skill` table and then removes the definition of the `skill` table from the database:

```
DROP TABLE skill
```

Like the **CREATE** statement, the **DROP** statement automatically executes a **COMMIT** before and after dropping the table. This makes permanent all changes to the database since the last **COMMIT** or **ROLLBACK**.

The **DROP** statement also drops all indexes on the table.

If you only want to remove data rows but not the table itself, use the **TRUNCATE TABLE** statement. If you truncate a table while other users are reading from it, the normal rules of table versioning apply, that is, old table versions remain until the transactions of the readers complete.

DROP TABLE and **TRUNCATE TABLE** statements execute in seconds. The size of the data does not affect the speed of the operation.

Creating Primary Keys

Each row in a table is uniquely identified by its primary key.

The **CREATE TABLE** and **ALTER TABLE** statements allow many attributes of tables to be set, including column constraints and checks.

Creating a Primary Key

The following statement creates the same `skill` table as before, except that a primary key is added:

```
CREATE TABLE skill (
  skill_id INTEGER NOT NULL,
  skill_name CHAR( 20 ) NOT NULL,
  skill_type CHAR( 20 ) NOT NULL,
  primary key( skill_id )
)
```

The primary key values must be unique for each row in the table which, in this case, means that you cannot have more than one row with a given `skill_id`.

Columns in the primary key are not allowed to contain NULL. You must specify **NOT NULL** on the column in the primary key.

Creating Foreign Keys

Each foreign key relationship relates a candidate key (primary key and unique constraint) in one column to a column in another table, which becomes the foreign key.

For example, you can create a table named `emp_skill`, which holds a description of each employee's skill level for each skill in which they are qualified, as follows:

```
CREATE TABLE emp_skill(
emp_id INTEGER NOT NULL,
skill_id INTEGER NOT NULL,
"skill level" INTEGER NOT NULL,
PRIMARY KEY( emp_id, skill_id ),
FOREIGN KEY REFERENCES employee,
FOREIGN KEY REFERENCES skill
)
```

The `emp_skill` table definition has a primary key that consists of two columns: the `emp_id` column and the `skill_id` column. An employee may have more than one skill, and so appear in several rows, and several employees may possess a given skill, so that the `skill_id` may appear several times.

The `emp_skill` table also has two foreign keys. The foreign key entries indicate that the `emp_id` column must contain a valid employee number that is a primary key in the `employee` table from the `employee` table, and that the `skill_id` must contain a valid entry that is a primary key in the `skill` table from the `skill` table.

A table can only have one primary key defined, but it may have as many foreign keys as necessary.

You cannot create foreign key constraints on temporary tables of any kind—local, global, or automatic.

For more information about valid strings and identifiers, see *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements*.

Table Information in System Views

Information about tables in a database is in the system view SYS.SYSTAB.

You can use the Sybase Control Center Execute SQL window or Interactive SQL to browse the information in this view. Type the following statement in the command window to see all the columns in the SYS.SYSTAB view:

```
SELECT *
FROM SYS.SYSTAB
```

Table Partitions

Table partitioning can improve performance by dividing 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.

Note: Table data is inaccessible if partitioned data exists in an offline dbspace. Ensure all dbspaces are online.

See also

- *Manage Data Storage* on page 119

Restrictions

Some restrictions apply to table partitions.

- Range-partitions and composite partitioning schemes, like hash-range partitions, require the separately licensed VLDB Management option.
- Only base tables can be partitioned; global temporary tables or declared local temporary tables cannot. All rows of a table partition are physically colocated.
- Although range partitions or subpartitions can reside in separate dbspaces, individual dbspaces are recommended for `BLOB` or `CLOB` columns or `CHAR`, `VARCHAR` or `VARBINARY` columns greater than 255 bytes only.

Partition Keys

Partition keys 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 key columns can contain NULL and DEFAULT values. All NULL values are mapped to the same partition.

You can have up to 8 partitioning key columns for hash partitions and only one key column for range partitions or subpartitions.

You can define up to 1024 range partitions or subpartitions.

DML Operations

You can perform DML operations including LOAD, INSERT, DELETE, TRUNCATE, and TRUNCATE TABLE PARTITION. UPDATE of partition key columns is not supported and results in an error. You can perform the UPDATE of all other columns of a partitioned table.

SAP Sybase IQ generates an exception for DML operations on a READ-ONLY table or READ-ONLY table partition. INSERT and LOAD statements or INSERT by updatable cursor generate an error and operations roll back, if the given row does not fit into the specified range of partitions.

DDL Operations

Some restrictions apply to DROP, RENAME, PARTITION, UNPARTITION, MERGE, SPLIT, and MOVE partitions:

Table 9. Restrictions on DDL Operations on Partitions

Operation	Restriction
Drop	You cannot drop a column from a partition key or range subpartition key. You cannot drop the last partition of a partitioned table.
Partition an unpartitioned table	An existing table cannot be made hash partitioned.
Merge two adjacent partitions	Both partitions must reside in the same dbspace. No data movement is required.
Split a partition	All rows must belong to the first of the two partitions after splitting. Split partition must be on same dbspace as original so that no data movement is required. You can split a range subpartition only if no data must be moved. All existing rows of the subpartition to be split must remain in the first subpartition after the split.

Operation	Restriction
Move a partition to a new dbspace.	All rows of the partition are moved to data pages in the new dbspace. `CREATE` permission in the new dbspace is required.

See also

- *Range Partitions* on page 165
- *Hash Partitions* on page 166
- *Hash-Range Partitions* on page 167
- *ALTER TABLE Statement* on page 426
- *CREATE TABLE Statement* on page 467

Range Partitions

Range partitioning divides large tables by a range of *partition-key* values established for each partition.

As part of an information life cycle management strategy, range partitioning can shorten backup and restore times; provide a finer level of granularity for data validation; and support tiered storage.

In a *range-partitioning-scheme*, the *partition-key* is the column whose value determines the partition that the row belongs to:

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

The *range-partition-declaration* determines how a named partition is placed in a dbspace:

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

The **VALUES** clause identifies the upper bound for each partition (in ascending order). Each range partition can be placed in its own dbspace and managed individually. Partition names must be unique within the set of partitions on a table.

Restrictions

Range partitioning is restricted to a single partition key column and a maximum of 1024 partitions.

Example

The most common form of range partitioning is to partition a table by date. This example creates 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);
```

See also

- *Restrictions* on page 163
- *Hash Partitions* on page 166
- *Hash-Range Partitions* on page 167
- *ALTER TABLE Statement* on page 426
- *CREATE TABLE Statement* on page 467

Hash Partitions

Hash partitioning maps data to partitions based on *partition-key* values processed by an internal hashing function.

Hash partitioning distributes data to logical partitions for parallel execution, which can enhance join performance on large tables and distributed queries (DQP).

In a *hash-partitioning-scheme* declaration, a *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:
    PARTITION BY HASH   ( partition-key [ , partition-key, … ] )
```

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.

Restrictions

- You cannot add, drop, merge, or split a hash partition.
- You cannot add or drop a column from a hash partition key.

Examples

In this example, table `tbl42` 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 )
  )
```

This example shows the common case where the join key is both the primary key and the hash partitioning key.

```
CREATE TABLE bar (
  c1 BIGINT NOT NULL,
  c2 CHAR(2) IQ UNIQUE(50),
  c3 DATE IQ UNIQUE(36524),
  c4 VARCHAR(200),
  c5 CLOB,
  PRIMARY KEY ( c1 )
  PARTITION BY HASH ( c1 )
)
```

See also

- *Restrictions* on page 163
- *Range Partitions* on page 165
- *Hash-Range Partitions* on page 167
- *ALTER TABLE Statement* on page 426
- *CREATE TABLE Statement* on page 467

Hash-Range Partitions

Hash-range partitioning is a composite partitioning scheme that subpartitions a hash-partitioned table by range.

Hash-range partitioning provides the benefits of hash partitioning and range partitioning. Hash partitioning provides the best distributed query performance; range sub-partitioning enhances administrative tasks as part of an information life cycle management strategy.

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.

Examples

This example (table `tbl42`) includes a `PRIMARY KEY` (column c1) and 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);
```

In this example, table `tbl43` includes different dbspaces for storage of `RANGE` subpartitions for column c4. This syntax is similar to existing syntax for `RANGE` partitions. Note the use of `SUBPARTITION` in column c4.

```
CREATE TABLE tbl43 (
   c1 bigint not null,
   c2 char(2) iq unique(50),
   c3 date
   c4 varchar(200)
   SUBPARTITION (P1 in Dsp331, P2 in Dsp332, P3 in Dsp333),
   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) ;
```

See also

- *Restrictions* on page 163
- *Range Partitions* on page 165
- *Hash Partitions* on page 166
- *ALTER TABLE Statement* on page 426
- *CREATE TABLE Statement* on page 467

Index SAP Sybase IQ Columns

SAP Sybase IQ indexes are designed to make the best use of memory, disk and CPU cycles for specific types of queries.

- Index sizes usually remain small. The entire database can be fully indexed and made available for ad hoc queries in the same space that would be needed to store the raw data. Traditional databases often need three or more times more space.
- Queries are resolved by efficiently combining and manipulating indexes on only the relevant columns. This avoids time-consuming table scans.
- I/O is minimized, eliminating potential bottlenecks.
- Because indexes are compact, more data can be kept in memory for subsequent queries, thereby speeding throughput on iterative analysis.
- Tuning is data dependent, allowing data to be optimized once for any number of ad hoc queries.

Overview of Indexes

Indexes improve data retrieval performance. SAP Sybase IQ indexes differ from traditional indexes and are designed to accelerate queries in the data warehousing environment.

When you load data into a table, SAP Sybase IQ physically stores data by column rather than by row, for each column in the table. The column orientation gives IQ indexes important advantages over traditional row-based indexing.

The default column storage structure that SAP Sybase IQ creates for each column is an index that is optimized for storing and projecting data. The column indexes you define are created as part of each individual table.

Logically, you can still access the data as in traditional row-based SQL databases. Column storage structures your data according to the attributes you want to track. In a data warehousing environment, you typically look at specific attributes of thousands or millions of data rows, rather than complete, single rows of data. Column storage optimizes your ability to perform selections or calculations on specific attributes.

Index Types Comparison

Choose the column index type appropriate to your task.

Table 10. Index Type Comparison

Index Type	Purpose
Compare (CMP) Index on page 172	Indexes relationship between any two distinct columns with identical data types, precision, and scale.
Containment (WD) Index on page 170	Stores words from a column string of `CHAR`, `VARCHAR`, or `LONG VARCHAR` data.
Date (DATE) and Datetime (DTTM) Indexes on page 173	Processes queries involving date or datetime quantities.
Fast Projection (FP) Index on page 177	Optimizes projections and enables certain kinds of search conditions to be evaluated. Assigned by default to any permanent table.
High_Group (HG) Index on page 178	Used for join columns with integer data types. Handles `GROUP BY` and equality operations efficiently. Recommended for columns with more than 1000 distinct values.
High_Non_Group (HNG) Index on page 180	Used for high-cardinality operations involving ranges or aggregates.
Low_Fast (LF) Index on page 181	Ideal for columns with under 1000 unique values, such as Yes/No, or number of dependents, or with `SUM`, `AVG`, and `COUNT`. Fastest index type in SAP Sybase IQ.
TEXT Index on page 182	Used for unstructured data analytics functionality, an SAP Sybase IQ licensed option.
Time (TIME) Index on page 183	Processes queries involving time quantities.

Containment (WD) Index

The Containment (**WD**) index allows you to store words from a column string of `CHAR`, `VARCHAR`, and `LONG VARCHAR` data.

Note: To create `LONG VARCHAR` columns, you must be specifically licensed to use the unstructured data analytics functionality. See *User-Defined Functions*.

Recommended Use of WD

Use a **WD** index for the fastest access to columns that contain a list of keywords (for example, in a bibliographic record or Web page).

These restrictions apply to **WD** indexes:

- You cannot specify the UNIQUE attribute.
- It can be used only with the CONTAINS or LIKE predicate.
- The column-name must identify a `CHAR`, `VARCHAR`, or `LONG VARCHAR` column in a base table.
- The minimum permitted column width is 3 bytes and the maximum permitted column width is the maximum width for a LOB column. (The maximum length is equal to 4GB multiplied by the database page size.)
- You must enclose the list of delimiters in single quotes.
- You omit the DELIMITED BY clause or specify the *separators-string* value as empty (single quotes), then SAP Sybase IQ uses the default set of separators. The default set of characters includes all 7-bit ASCII characters that are not 7-bit ASCII alphanumeric characters, except the hyphen and the single quotation mark, which, by default, are part of words. There are 64 separators in the default separator set.
- If you specify multiple DELIMITED BY and LIMIT clauses, no error is returned, but only the last clause of each type is used. For example, the following two statements return identical results:

  Statement 1:

  ```
  CREATE WD INDEX c1wd on foo(c1)
  DELIMITED BY 'f' LIMIT 40 LIMIT 99 DELIMITED BY 'g' DELIMITED BY
  'h';
  ```

  Statement 2:

  ```
  CREATE WD INDEX c1wd on foo(c1)
  DELIMITED BY 'h' LIMIT 99;
  ```

- 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 may return misleading results. For example, column `company_name` contains two words delimited by a separator, but the second word is blank padded:

  ```
  'Concord' 'Farms                                                '
  ```

  Suppose that a user entered the following query:

  ```
  SELECT COUNT(*)FROM Customers WHERE CompanyName contains ('Farms')
  ```

  The parser determines that the string contains:

  ```
  'Farms                                                  '
  ```

  instead of:

```
'Farms'
```

and returns 1. You can avoid this problem by using VARCHAR instead of CHAR columns.

- The **sp_iqcheckdb** (DBCC consistency checker) allocation, check, verify, and repair modes support the **WD** index on CHAR, VARCHAR, and LONG VARCHAR columns.

Advantages and Disadvantages of WD

Advantages and disadvantages of using a **WD** index.

Table 11. WD Advantages and Disadvantages

Advantages	Disadvantages
Huge performance gains are possible for large loads. Certain LIKE predicates execute faster with this index. CONTAINS predicate used with this index takes precedence over the LIKE predicate. Best way to index keywords or parts of a URL.	Disk space requirements may potentially be very large. Index not possible if uniqueness enforced. Can use this index only if data in your columns is CHAR, VARCHAR, or LONG VARCHAR.

Compare (CMP) Index

A Compare (CMP) index is the relationship between two columns. You may create Compare indexes on any two distinct columns with identical data types, precision, and scale. The **CMP** index stores the binary comparison (<, >, or =) of its two columns.

Recommended Use of CMP

You can create a **CMP** index on columns that are NULL, NOT NULL, or a mixture. The **CMP** index cannot be unique.

Numeric and decimal data types are considered identical. You may create **CMP** indexes on them when precision and scale are identical. For CHAR, VARCHAR, BINARY, and VARBINARY columns, precision means having the same column width.

For example, these commands create a table, then create appropriate **CMP** indexes:

```
CREATE TABLE f(c1 INT NOT NULL, c2 INT NULL, c3 CHAR(5), c4 CHAR(5))
```

```
CREATE CMP INDEX c1c2cmp ON f(c1, c2)
```

This index is illegal because the columns indexed are not of the same data type, precision, and scale:

```
CREATE CMP INDEX c1c3cmp ON f(c1, c3)
```

Restrictions on CMP Indexes

The following restrictions apply to **CMP**.

- You can drop **CMP** indexes.
- **CMP** indexes cannot be unique.
- An exception is raised if you attempt to alter or delete a column that is defined in a **CMP** index.
- You cannot **ALTER TABLE MODIFY** an existing column that is defined in a **CMP** index.
- **CMP** indexes do not support the `BIT`, `FLOAT`, `DOUBLE`, and `REAL` data types.

Date (DATE) and Datetime (DTTM) Indexes

DATE and **DTTM** index types process queries involving date or datetime quantities.

- A **DATE** index is used on columns of data type `DATE` to process certain queries involving date quantities.
- The **DTTM** index is used on columns of data type `DATETIME` or `TIMESTAMP` to process certain queries involving datetime quantities.

Recommended Use of DATE and DTTM Index Types

Use a **DATE** or **DTTM** index when the `DATE`, `DATETIME`, or `TIMESTAMP` column is used in queries containing date and time functions and operations.

- Queries with DATEPART equality predicates (=, !=), DATEPART range predicates (>, <, >=, <=, !>, !<, **BETWEEN**) and DATEPART IN list predicates
- Queries with range predicates (>, <, >=, <=, **BETWEEN**)

Note: For a simple equality predicate (no DATEPART) with a `DATE`, `DATETIME`, or `TIMESTAMP` column, **LF** and **HG** indexes provide the best performance. If an **LF** or **HG** index is unavailable, the **DATE** or **DTTM** index is used to get the result.

If a `DATE`, `DATETIME`, or `TIMESTAMP` column is used in the **GROUP BY** clause or in the **WHERE/HAVING** clauses for equalities (including join conditions) or **IN** predicates, the column needs an **LF** or **HG** index, as only these indexes can perform fast equality.

The table `tab` used in the examples contains columns defined as follows:

```
CREATE TABLE tab
(col1 DATE,
 col2 DATETIME,
 col3 TIME);
```

Queries with DATEPART Equality, Range, and IN List Predicates

For a query with an equality predicate (= or !=), if one side of the comparison is a DATEPART expression or some other date and time function (for example, YEAR, QUARTER, DAY, MINUTE), and the other side of the comparison is a constant expression (including a constant

value or host variable), then the **DATE**, **TIME**, or **DTTM** index is used (if the index is available) to get the result set.

For example, the **DATE**, **TIME**, or **DTTM** index is used in these queries:

```
SELECT * FROM tab WHERE DATEPART(YEAR, col1) = 2002;

SELECT * FROM tab WHERE DATEPART(HOUR, col2) = 20;

SELECT * FROM tab WHERE MINUTE (col3) != 30;

SELECT * FROM tab WHERE DATEPART(MONTH, col2) = @tmon;
```

where @tmon is an INTEGER host variable.

The appropriate DATEPART range and IN list predicate conditions for processing with **DATE**, **TIME**, and **DTTM** indexes are:

- COMPARISON conditions >, <, >=, <=, !>, !<
  One side of the operator is a date/time function or DATEPART function, with a parameter that is a table column or view column. The other side of the operator is a constant expression, such as an integer or integer type host variable. For example:

  ```
  DATEPART(WEEK, col1) !<23

  DATEPART(YEAR, col1) = 2001

  HOUR(col3) >= 1
  ```
- BETWEEN ... AND condition
  The left side of BETWEEN is a date/time function or DATEPART function, with a parameter that is a table column or view column. Both sides of the AND are constant expressions, such as integers or integer type host variables. For example:

  ```
  DATEPART(YEAR, col1) BETWEEN host-var1 AND host-var2
  ```
- IN conditions
  The left side of IN is a date/time function or DATEPART function, with a parameter that is a table column or view column. The values inside the IN list are constant expressions. For example:

  ```
  DATEPART(MONTH, col1) IN (1999, 2001, 2003)
  ```

Note: The **DATE**, **TIME**, and **DTTM** indexes do not support some date parts (Calyearofweek, Calweekofyear, Caldayofweek, Dayofyear, Millisecond). For example:

```
SELECT * FROM tab WHERE DATEPART(MILLISECOND, col3)
= 100;

SELECT * FROM tab WHERE DATEPART(DAYOFYEAR, col1) <= 89;
```

In these cases, the query optimizer chooses other indexes to get the result.

Queries with Range Predicates

Examine how your queries use range predicates when indexing predicate columns.

In the following cases with range predicates, a **DATE**, **TIME**, or **DTTM** index is chosen to process the queries:

- Compare condition:

```
SELECT * FROM tab WHERE col1 < '2002/10/09';

SELECT * FROM tab WHERE col2 >= '2002/01/01 09:12:04.006';
```

  One side of the comparison operator is a column name and the other side is a constant expression (constant value or host variable).
- Between condition:

```
SELECT * FROM tab WHERE col3 BETWEEN '09:12:04.006' AND
'20:12:04.006';

SELECT * FROM tab WHERE col2 BETWEEN tmp_datetime1 AND
tmp_datetime2;
```

For these types of queries, a **DATE**, **TIME**, or **DTTM** index is usually faster than a **HNG** index.

In three specific cases, use of the **DATE** or **DTTM** index may significantly improve performance:

- The range of the predicate is exactly one or more years (the actual start date is the beginning of a year and the actual end date is the end of a year). For example:

```
SELECT * FROM tab WHERE col1 BETWEEN '1993-01-01' AND
'1996-12-31';

SELECT * FROM tab WHERE col1 >= '1993-01-01' AND
col1 < '1997-01-01';

SELECT * FROM tab WHERE col2 BETWEEN '1993-01-01 00:00:00.000000'
AND '1996-12-31 23:59:59.999999';
```

- The range of the predicate is exactly one or more months in the same year (the actual start date is the beginning of a month and the actual end date is the end of a month). For example,

```
SELECT * FROM tab WHERE col1 > '1993-01-31' AND
col1 <= '1993-06-31';

SELECT * FROM tab WHERE col2 >= '1993-01-01 00:00:00.000000' AND
col1 < '1993-06-01 00:00:00.000000';
```

- The range of the predicate is exactly one day. For example,

```
SELECT * FROM tab WHERE col2 >= '1993-01-31 00:00:00.000000' AND
col2 <= '1993-01-31 23:59:59.999999';
```

Note: In the three cases above, you must be careful about the concepts of range of years, range of months, and exactly one day. For example, there are four cases for a **DTTM** index that are recognized as range of years:

```
col2 >  'year1/12/31 23:59:59.999999' and
col2 <  'year2/01/01 00:00:00.000000'
```

```
col2 >= 'year1/01/01 00:00:00.000000' and
col2 <  'year2/01/01 00:00:00.000000'

col2 >  'year1/12/31 23:59:59.999999' and
col2 <= 'year2/12/31 23:59:59.999999'

col2 >= 'year1/01/01 00:00:00.000000' and
col2 <= 'year2/12/31 23:59:59.999999'
```

Ranges as in the following examples do not match range of years:

```
col2 > 'year1/12/31 23:59:59.999999' and
col2 <= 'year2/01/01 00:00:00.000000'

col2 > 'year1/01/01 00:00:00.000000' and
col2 <  'year2/01/01 00:00:00.000000'
```

The first range does not match, because it includes the value 'year2/01/01 00:00:00:000000' in addition to the range of years. The second range loses the value 'year1/01/01 00:00:00.000000.'

Similar specifics apply to range of months, and exactly one day, for both **DTTM** and **DATE** indexes.

If a small date range (less than 60 values) does not fit the three specific cases above, then **LF** and **HG** indexes are faster than the **DATE** index.

Advantages and Disadvantages of DATE/DTTM

Advantages and disadvantages of using a **DATE** or **DTTM** index.

Table 12. DATE/TIME/DTTM Advantages and Disadvantages

Advantages	Disadvantages
Queries with date, time, or datetime quantities are resolved more quickly than with other index types. You can create and drop a **DATE** or **DTTM** index.	Uses more disk space than **HNG** index. Fast equality still requires LF or HG index. You can use these indexes only if data in the column is `DATE`, `DATETIME`, or `TIMESTAMP` data type.

Restrictions on DATE/DTTM Indexes

Restrictions apply to **DATE** and **DTTM** indexes:

- Cannot use the UNIQUE keyword.
- Can only be created on a single column.
- Do not support date parts Calyearofweek, Calweekofyear, Caldayofweek, Dayofyear, Millisecond.

Comparison to Other Indexes

The **DATE**, **TIME**, and **DTTM** indexes have performance consistent with the **HNG** index.

Compared to **HNG**, **DATE**, **TIME**, and **DTTM** indexes are generally up to twice as fast as **HNG** in the supported cases. Therefore, an **HNG** index is not necessary in addition to a **DATE**, **TIME**, or **DTTM** index on a column of `DATE`, `TIME`, `DATETIME`, or `TIMESTAMP` data type.

Additional Indexes for DATE/DTTM Columns

Always use a **DATE**, **TIME**, or **DTTM** index on a column of `DATE`, `TIME`, `DATETIME`, or `TIMESTAMP` data type, if the column is referenced in the WHERE clause, in ON conditions, or in the GROUP BY clause.

In addition, the **HG** or **LF** index may also be appropriate for a `DATE`, `TIME`, `DATETIME`, or `TIMESTAMP` column, especially if you are evaluating equality predicates against the column. A **LF** index is also recommended, if you frequently use the column in the GROUP BY clause and there are fewer than 1000 distinct values (that is, less than three years of dates).

Fast Projection (FP) Index

By default, the database engine automatically creates a Fast Projection (**FP**) index on all columns. **FP** indexes optimize projections and allow the database engine to evaluate certain kinds of search conditions.

An `FP` index is an array of *n* fixed-length entries where *n* is the number of rows in the table. The IQ UNIQUE storage directive applied to the column determine whether the column loads as a flat `FP` or `NBit`. `Flat FP` indexes contain actual column cell values. `NBit` is a compression scheme that uses *n* bits to index the dictionary where the data is stored. All datatypes except `LOB` (both character and binary) and `BIT` datatypes may be `NBit` columns.

IQ UNIQUE

IQ UNIQUE defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit. 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 and FP_NBIT_LOOKUP_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.

Rebuilding FP Column Indexes

sp_iqindexmetadata displays details about column indexes, including the `FP` index type (`Flat FP` or `NBIT`), the distinct counts, `IQ UNIQUE` *n* value, and dictionary size.

sp_iqrebuildindex rebuilds `Flat FP` as `NBIT`, or `NBIT` as `Flat FP`. The `index_clause` can reset `IQ UNIQUE` *n* to an explicit value from '0' (to recast an `NBIT` column to `Flat FP`) up to the limits defined in the `FP_NBIT_AUTOSIZE_LIMIT` and `FP_NBIT_LOOKUP_MB` options. If the count exceeds the *n* value, and `FP_NBIT_ENFORCE_LIMITS=ON`, the operation rolls back. If the `FP_NBIT_ENFORCE_LIMITS=OFF` (default), the `NBIT` dictionary continues to grow.

Additional Information

- *Reference: Statements and Options > Database Options > Alphabetical List of Options > FP_NBIT_AUTO_LIMIT*
- *Reference: Statements and Options > Database Options > Alphabetical List of Options > FP_NBIT_LOOKUP_MB*
- *RReference: Statements and Options > Database Options > Alphabetical List of Options > FP_NBIT_ROLLOVER_MAX_MB*
- *Reference: Statements and Options > Database Options > Alphabetical List of Options > FP_NBIT_ENFORCE_LIMIT*
- *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*

High_Group (HG) Index

The High_Group index is commonly used for join columns with integer data types. It is also more commonly used than High_Non_Group because it handles **GROUP BY** efficiently.

Recommended Use of HG

HG indexes and multicolumn HG indexes offer different functionality that must be considered when choosing which to use.

Use an **HG** index when:

- The column will be used in a join predicate
- A column has more than 1000 unique values

Use multicolumn **HG** indexes to enhance the performance of **ORDER BY** queries with references to multiple columns. This change is transparent to users, but improves query performance.

Note: Foreign key columns require their own, individual **HG** index.

Advantages and Disadvantages of High_Group

High_Group indexes have advantages and disadvantages.

Table 13. HG Advantages and Disadvantages

Advantages	Disadvantages
Quickly processes queries with **GROUP BY**.	This index needs more disk space than the **HNG** index (it can take as much as three times more space than raw data). This index type takes the longest time to populate with data, and to delete. You cannot use this index if data in your columns is `BIT`, `VARBINARY` > 255 bytes, **CHAR** > 255 bytes, or `VARCHAR` > 255 bytes. This index is not recommended for `FLOAT`, `REAL`, and `DOUBLE` data.

Comparison with Other Indexes

A comparison of HG with LF and HNG indexes.

LF

The determining factor is the number of unique values. Use High_Group if the number of unique values for the column is high. Use Low_Fast if the number of unique values is low.

HNG

The determining factor is whether the column is a join column, and/or whether **GROUP BY** may be processed on the column. If either of these is true, use High_Group, either alone or in combination with High_Non_Group. Otherwise, use High_Non_Group to save disk space.

Additional Indexes for HG Columns

In some cases, a column that meets the criteria for a High_Group index may be used in queries where a different type of index may be faster. If this is the case, create additional indexes for that column.

Automatic Creation of High_Group Index

SAP Sybase IQ creates a High_Group index by default whenever you issue a **CREATE INDEX** statement without specifying an index type.

SAP Sybase IQ automatically creates a High_Group index for any **UNIQUE**, **FOREIGN KEY**, or **PRIMARY KEY** constraint. For foreign keys of a single column, SAP Sybase IQ creates a single column non-unique High_Group index. For multicolumn foreign keys, a non-unique composite High_Group index is implicitly created. The non-unique HG index allows

duplicate values and optionally allows nulls. It provides the building block for referential integrity and can be used to improve query performance.

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. For more details on Multicolumn Indexes, see *CREATE INDEX Statement* in *Reference: Statements and Options*.

Queries with joins on multiple columns or multicolumn group by clauses may improve performance because a non-unique composite High Group index provides more accurate cardinality estimates of joins and result sizes. It can also optimize pushdowns and subqueries.

High_Non_Group (HNG) Index

Add an **HNG** index for range searches.

An **HNG** index requires approximately three times less disk space than an **HG** index. On that basis alone, if you do not need to do group operations, use an **HNG** index instead of an **HG** index.

Conversely, if you know you are going to do queries that a **HG** index handles more efficiently, or if the column is part of a join, or you want to enforce uniqueness, use an **HG** index.

Note: Using the **HNG** index instead of the **HG** index may seriously degrade performance of complex ad hoc queries joining four or more tables. If query performance is important for such queries in your application, choose both **HG** and **HNG**.

Recommended Use of HNG

The **HNG** index is not recommended for **GROUP BY** queries..

Use an **HNG** index when:

- The number of unique values is high (greater than 1000)
- You don't need to do **GROUP BY** on the column

Advantages and Disadvantages of High_Non_Group

High_Non_Group indexes have advantages and disadvantages.

Table 14. HNG advantages/disadvantages

Advantages	Disadvantages
Due to compression algorithms used, disk space requirements can be reduced without sacrificing performance. If the column has a high number of unique values, this is the fastest index, with a few exceptions.	This index type is not recommended for **GROUP BY** queries. Impossible if uniqueness is enforced. You cannot use this index type if data in your columns is `FLOAT`, `REAL`, `DOUBLE`, `BIT`, `BINARY`, `VARBINARY`, **CHAR** > 255 bytes, or `VARCHAR` > 255 bytes.

Comparison to Other Indexes

A comparison of the HNG index to similar indexes.

- **HNG** needs less disk space than **HG** but cannot perform **GROUP BY** efficiently.
- In choosing between **LF** and **HNG**, the determining factor is the number of unique values. Use **HNG** when the number of unique values is greater than 1000.

Additional Indexes for HNG Columns

The High_Group index is also appropriate for an **HNG** column.

Low_Fast (LF) Index

This index is ideal for columns that have a very low number of unique values (under 1,000) such as sex, Yes/No, True/False, number of dependents, wage class, and so on. **LF** is the fastest index in SAP Sybase IQ.

When you test for equality, just one lookup quickly gives the result set. To test for inequality, you may need to examine a few more lookups. Calculations such as **SUM**, **AVG**, and **COUNT** are also very fast with this index.

As the number of unique values in a column increases, performance starts to degrade and memory and disk requirements start to increase for insertions and some queries. When doing equality tests, though, it is still the fastest index, even for columns with many unique values.

Recommended Use

Use an **LF** index when a column has fewer than 1,000 unique values, or a column has fewer than 1,000 unique values and is used in a join predicate.

Never use an **LF** index for a column with 10,000 or more unique values. If the table has fewer than 25,000 rows, use an **HG** index, as fewer disk I/O operations are required for the same operation.

Advantages and Disadvantages of Low_Fast

Low_Fast indexes have advantages and disadvantages.

Advantages	Disadvantages
This index is fast, especially for single table **SUM**, **AVG**, **COUNT**, **COUNT DISTINCT**, **MIN**, and **MAX** operations.	Use this index type for a maximum of 10,000 unique values. You cannot use this index type if data in your columns is `BIT`, `VARBINARY` > 255 bytes, **CHAR** > 255 bytes, or `VARCHAR` > 255 bytes.

Comparison with Other Indexes

Familiarize yourself with how the LF index compares to other indexes.

HNG/HG

The main factor to consider is the number of unique values within a column. Use **LF** if the number is low.

Additional Indexes

The **High_Non_Group** index type may be appropriate for a **Low_Fast** column.

Note: It is almost always best to use an **LF** index if the number of unique values is low (less than 1,000). Consider this index first, if the column appears in the **WHERE** clause. Consider other index types (**HG** and **HNG**) only when the number of unique values is high. For range queries with a high number of unique values, also consider having an **HNG** index.

TEXT Index

To use **TEXT** indexes, you must be specifically licensed to use the unstructured data analytics functionality.

Note: See Unstructured Data Analytics.

Unlike a Containment (WD) index, which uses keywords in a column string, a **TEXT** index stores positional information for terms in the indexed columns. Queries that use **TEXT** indexes can be faster than those that must scan all the values in the table.

Creating a TEXT Index Using Interactive SQL

Before you can perform a full text search, you must create a **TEXT** index on the columns you want to search.

A **TEXT** index stores positional information for terms in the indexed columns.

1. Connect to the database as a user with CREATE ANY INDEX or CREATE ANY OBJECT system privilege, or as the owner of the underlying table.
2. Execute a **CREATE TEXT INDEX** statement.

This example creates a **TEXT** index, `myTxtIdx`, on the `CompanyName` column of the `Customers` table in the `iqdemo` database. The `default_char` text configuration object is used.

```
CREATE TEXT INDEX myTxtIdx ON Customers
    ( CompanyName ) CONFIGURATION default_char
```

Time (TIME) Index

Use the TIME index to process queries involving time quantities.

The **TIME** index is used on columns of data type `TIME`.

Recommended Use of TIME Index Type

Use a **TIME** when the `TIME` column is used in queries containing time functions and operations.

- Queries with DATEPART equality predicates (=, !=), DATEPART range predicates (>, <, >=, <=, !>, !<, **BETWEEN**) and DATEPART IN list predicates
- Queries with range predicates (>, <, >=, <=, **BETWEEN**)

Note: For a simple equality predicate (no DATEPART) with a `TIME` column, **LF** and **HG** indexes have the best performance. If an **LF** or **HG** index is unavailable, the**TIME** index gets the result.

If a `TIME` or `TIMESTAMP` column is used in the GROUP BY clause or in the WHERE/HAVING clauses for equalities (including join conditions) or IN predicates, the column needs an **LF** or **HG** index, as only these indexes can do fast equality.

Advantages and Disadvantages of TIME

TIME indexes have advantages and disadvantages.

Advantages	Disadvantages
Queries with time quantities are resolved more quickly than with other index types. You can create and drop a **TIME** index.	Uses more disk space than **HNG** index. Fast equality still requires LF or HG index. You can use this index only if data in the column is `TIME` data type.

Restrictions on TIME Indexes

The following restrictions currently apply to **TIME** indexes:

- Cannot use the UNIQUE keyword.
- Can only be created on a single column.
- Does not support date part Millisecond.

Criteria for Choosing Indexes

The set of indexes that you define for any given column can have dramatic impact on the speed of query processing.

There are four main criteria for choosing indexes:

- Number of unique values
- Types of queries
- Disk space usage
- Data types

Use the recommendations for all criteria in combination, rather than individually. Remember also that all columns are automatically stored in a way that facilitates fast projections. To decide on additional indexes, look closely at the data in each column. Try to anticipate the number of unique and total values expected in query results.

Number of Unique Values in the Index

Indexes are optimized according to the number of unique (distinct) values they include.

IQ UNIQUE defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit. 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 and FP_NBIT_LOOKUP_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.

When the number of distinct values reaches certain levels, choose indexes according to the recommendations in this table.

Table 15. Consideration Order

Number of Unique Values	Recommended Index Type
Below 1,000	**LF** (**HG** if table has <100,000 rows)
1000 and over	**HG** and/or **HNG**

Here are some examples of columns with different numbers of unique values:

- Columns that hold marital status have just a few unique values (single, married, NULL)
- Columns that hold state or province names have fewer than 100 unique values
- Columns that hold date data probably have more than 100 but fewer than 65536 unique values
- Columns that hold account numbers or social security numbers may have thousands or millions of unique numbers

Types of Queries

Certain index types optimize particular types of queries.

Determine in advance how data in the columns will generally be queried. For example:

- Will the column be part of a join predicate?
- If the column has a high number of unique values, will the column be used in a **GROUP BY** clause, be the argument of a **COUNT DISTINCT**, or be in the **SELECT DISTINCT** projection?
- Will the column frequently be compared with another column of the same data type, precision, and scale?

Often, the type of data in a column gives a good indication how the column will be used. For example, a date column will probably be used for range searches in **WHERE** clauses, and a column that contains prices or sales amounts will probably be used in the projection as an argument for aggregate functions (**SUM**, **AVG**, and so on).

Note: SAP Sybase IQ can still resolve queries involving a column indexed with the wrong index type, although it may not do so as efficiently.

For optimal query performance, columns used in join predicates, subquery predicates, GROUP BY and DISTINCT clauses should have either a **HG** or **LF** index, since IQ has no statistics other than the index for the optimizer to use. Use **HG** for high cardinality and **LF** for low cardinality columns, except for tables with fewer than 100,000 rows which should have **HG**.

These estimates are generally valid; however, other factors can take precedence:

- For range predicates, the number of unique values is the most important factor.
- When using the set functions **COUNT**, **COUNT DISTINCT**, **SUM**, **MIN**, **MAX**, and **AVG**, to use any index other than the default, the entire query must be resolvable using a single table.

- `BIT` data can be used only in the default index; `VARBINARY` data greater than 255 bytes can be used only in the default, **TEXT**, and **CMP** index types; `CHAR` and `VARCHAR` data greater than 255 bytes can be used only in the default, **CMP**, **TEXT**, and **WD** index types; **LONG VARCHAR** data can be used only in the default, **TEXT**, and **WD** index types; only `DATE` data can be used in the **DATE** index type; only `TIME` data can be used in the **TIME** index type; only `DATETIME` and `TIMESTAMP` data can be used in the **DTTM** index type.

Indexes Recommended by Query Type

Use the index type recommended for your query.

The index that is usually fastest for each query is listed first, the slowest last. These recommendations should not be your only criteria for choosing an index type; consider the number of unique values and disk space.

Type of Query Usage	Recommended Index Type
In a **SELECT** projection list	Default
In calculation expressions such as **SUM**(A+B)	Default
As **AVG/SUM** argument	**LF, HG**, default
As **MIN/MAX** argument	**LF, HG**
As **COUNT** argument	Default
As **COUNT DISTINCT, SELECT DISTINCT** or **GROUP BY** argument	**LF**, **HG**, default
As analytical function argument	**LF**, default
If field does not allow duplicates	**HG**
Columns used in ad hoc join condition	Default, **HG, LF**,
As **LIKE** argument in a **WHERE** clause	Default
As **IN** argument	**HG, LF**
In equality or inequality (=, !=)	**HG, LF**; also **CMP**
In range predicate in **WHERE** clause (>, <, >=, <=, **BETWEEN**	**LF** or **HG**; also **CMP**, **DATE**, **TIME**, **DTTM**
In **DATEPART** equality, range, and IN list predicates	**DATE**, **TIME**, **DTTM**
In a **CONTAINS** predicate	**WD**, **TEXT**

Indexes Recommended by Column Value

Criteria such as the number of unique values in a column help identify appropriate index types for data.

Criteria to identify	Index to select
Note indexes created automatically on all columns.	Default index
Note indexes created automatically on columns with **UNIQUE** or **PRIMARY KEY** constraint.	**HG** with **UNIQUE** enforced
Identify all columns used in a join predicate and choose the index type depending on the number of unique values.	**HG** or **LF**
Identify columns that contain a low number of unique values and do not already use multiple indexes.	**LF**
Identify columns that have a high number of unique values and that are part of a **GROUP BY** clause in a select list in a **SELECT DISTINCT** or **DISTINCT COUNT**.	**HG**
Identify columns that may be used in the **WHERE** clause of ad hoc join queries that do not already have **HG** or **LF** indexes.	**HG** or **LF**
Identify columns that have a high number of unique values and that will not be used with **GROUP BY**, **SELECT DISTINCT** or **DISTINCT COUNT**.	**HNG**
Identify pairs of columns with the same data type, precision, and scale that are likely to need frequent comparison.	**CMP**
Identify columns that contain a list of keywords or a URL.	**WD**
Identify columns of `DATE`, `TIME`, `DATETIME`, or `TIMESTAMP` that have a high number of unique values and that will not be used with **GROUP BY**, **SELECT DISTINCT**, or **DISTINCT COUNT**.	**DATE**, **TIME**, or **DTTM**
Look at any remaining columns and decide on additional indexes based on the number of unique values, type of query, and disk space. Also, for all columns, be sure that the index types you select allow the data type for that column.	

Disk Space Usage

The following table provides estimates of the amount of space each index uses compared to the amount of column data from the source database or flat file.

Table 16. Index disk space usage

Type of index	Estimated space versus raw data	Comments
Default	Smaller than or equal to	If the number of distinct values is less than 255, this index uses significantly less space than the raw data
High_Group	Smaller than up to twice as large	As the number of distinct values decreases (that is, the number of entries per group increases), the space used decreases in proportion to the size of the raw data
High_Non_Group	Smaller than or equal to	Smaller than the raw data in most cases
Low_Fast	Smaller than up to twice as large	Same as High_Group
Date	Smaller than or equal to	Larger than High_Non_Group
Time	Smaller than or equal to	Larger than High_Non_Group
Datetime	Smaller than or equal to	Larger than High_Non_Group

For **LF** and **HG** indexes, the index size depends on the number of unique values. The more unique values, the more space the index requires.

Because **CMP** indexes are always an additional index, they do not save disk space.

Data Types in the Index

When indexing a column, choose only supported index types for the column data type.

The default index allows any data type.

Table 17. Indexes Supported for Data Types

Data Type	Supported Indexes	Unsupported Indexes
tinyint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT
smallint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT
int	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT
unsigned int	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT
bigint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT

Data Type	Supported Indexes	Unsupported Indexes
unsigned bigint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT
numeric, decimal	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM, TEXT
double	LF (HG permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM, TEXT
float	LF (HG permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM, TEXT
real	LF (HG permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM, TEXT
bit	(Default index only)	CMP, HG, HNG, LF, WD, DATE, TIME, DTTM, TEXT
date	CMP, HG, HNG, LF, DATE	WD, TIME, DTTM, TEXT
time	CMP, HG, HNG, LF, TIME	WD, DATE, DTTM, TEXT
datetime, timestamp	CMP, HG, HNG, LF, DTTM	WD, DATE, TIME, TEXT
char <= 255 bytes, character	CMP, HG, HNG, LF, WD, TEXT	DATE, TIME, DTTM
char >255 bytes	CMP, WD, TEXT	HG, HNG, LF, DATE, TIME, DTTM
varchar <= 255 bytes	CMP, HG, HNG, LF, WD, TEXT	DATE, TIME, DTTM
varchar >255 bytes	CMP, WD, TEXT	HG, HNG, LF, DATE, TIME, DTTM
long varchar	WD, TEXT	CMP, HG, HNG, LF, DATE, TIME, DTTM
binary <= 255 bytes	CMP, HG, LF, TEXT	HNG, WD, DATE, TIME, DTTM
binary > 255 bytes	CMP, TEXT	HG, HNG, LF, WD, DATE, TIME, DTTM
varbinary <= 255 bytes	CMP, HG, LF, TEXT	HNG, WD, DATE, TIME, DTTM
varbinary > 255 bytes	CMP, TEXT	HG, HNG, LF, WD, DATE, TIME, DTTM

Index Type Combinations

When a column is used in more than one type of query, you might want to assign more than one column index type.

Table 18. Index Combinations

	Index to Add					
Existing Index	HG	HNG	LF	CMP[a]	WD	DATE, TIME, or DTTM
HG	-	1	2	1	1	1
HNG	1	-	1	1	2	2
LF	2	1	-	1	2	1

1 = Combination recommended.

2 = Combination not recommended.

a. A CMP index applies to a pair of columns. Each of those columns always has at least one other index.

Creating Indexes

SAP Sybase IQ automatically creates a default index on every table column when you create a database in an IQ store. For best query performance, create additional indexes.

Prerequisites

If you are creating an index on a partitioned table, all dbspaces must be online.

Task

1. Choose the right index type for your data. Make sure that the selected index type allows the data type for that column.

   If you do not specify an index type, SAP Sybase IQ creates an **HG** index. Several front-end tools create an **HG** index automatically for this reason.
2. Create an index using any method:

   - Enter a **CREATE INDEX** statement.

     For example, create a High_Non_Group (**HNG**) index called `ShipIx` on the `ShipDate` column of the `SalesOrderItems` table.

     ```
     CREATE HNG INDEX ShipIx
       ON dbo.SalesOrderItems (ShipDate)
     ```
   - Use the Sybase Control Center Create Index wizard.

- Use a PRIMARY KEY, or UNIQUE column constraint of **CREATE TABLE**, which creates an **HG** index automatically.

See also

- *CREATE INDEX Statement* on page 455
- *CREATE TABLE Statement* on page 467

Concurrent Column Indexing

In some cases, you can create more than one column index simultaneously.

- Each **CREATE INDEX** statement can create only one index.
- If two connections issue **CREATE INDEX** statements on the same table, the first statement works; the second receives an error stating that only one writer is allowed.
- If two connections issue **CREATE INDEX** statements on different tables, both proceed in parallel.

Status Messages for Index Loading

If you enable progress messages, status messages display after every 100,000 records are inserted.

To enable messages, set the **NOTIFY** option of **CREATE INDEX** to a non-zero value.

To change the number of records, use the **NOTIFY** option of **CREATE INDEX**, or the option **NOTIFY_MODULUS**. To prevent these messages, specify **NOTIFY 0**.

Adding Column Indexes After Inserting Data

When you create an additional column index, the **CREATE INDEX** command creates the new index as part of the individual table.

If the existing column indexes in the individual table already contain data, the **CREATE INDEX** statement also inserts data into the new index from an existing index. This ensures data integrity among all the column indexes for columns within an individual table.

This capability is useful if you discover that a column needs an additional index after you have already inserted data. This allows you to add the index without having to start over.

Note: Inserting data from an existing index can be slow. It is always faster to create all the appropriate indexes before you insert data, then insert into all of them at once, with either the **LOAD TABLE** or **INSERT** statement.

Executing Groups of CREATE INDEX Statements

Use the keywords **BEGIN PARALLEL IQ** and **END PARALLEL IQ** to delimit any number of **CREATE INDEX** statements that you want to execute as a group at the same time.

Use these keywords only to create indexes on base tables in the IQ main store, not temporary tables or catalog store tables.

If any of these **CREATE INDEX** statements fails, they all roll 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
```

Running the Index Advisor

If you set the INDEX_ADVISOR option on your database, SAP Sybase IQ issues messages in the message log or query plan to suggest additional indexes that might improve performance.

1. In Interactive SQL, set the **INDEX_ADVISOR** option:
   ```
   SET OPTION index_advisor = 'ON'
   ```
2. Examine the message log file. By default, the message log is named `dbname.iqmsg`, and the file is created in the same directory as the catalog store.

   Look for messages that focus on the following areas:
   - Local predicate columns
   - Single-column join key columns
   - Correlated subquery columns
   - Grouping columns
3. If you decide to follow the recommendations of the index advisor messages, create the suggested indexes:
   ```
   CREATE HG INDEX id_hg
   ON SalesOrderItems
   ( ID ) IN Dsp5
   ```

Renaming Indexes

You can rename an index in a base table or global temporary table with the owner type USER. To rename an index, use the ALTER INDEX statement.

Note: You cannot rename indexes created to enforce key constraints.

Viewing Indexes

Use the stored procedure **sp_iqindex** to see information about indexes.
To list the indexes in the `Departments` table, issue:

- ```
  sp_iqindex 'Departments'
  ```

  For readability, output is shown here in two pieces:
  ```

table_name	table_owner	column_name	index_type
Departments	GROUPO	DepartmentHeadID	FP
Departments	GROUPO	DepartmentHeadID	HG
Departments	GROUPO	DepartmentID	FP
Departments	GROUPO	DepartmentID	HG
Departments	GROUPO	DepartmentName	FP

- To display information for all tables, omit the table name from **sp_iqindex**:

index_name	unique_index	dbspace_id	remarks
ASIQ_IDX_T740_C3_FP	N	16,387	(NULL)
ASIQ_IDX_T740_C3_HG	N	16,387	(NULL)
ASIQ_IDX_T740_C1_FP	U	16,387	(NULL)
ASIQ_IDX_T740_I4_HG	Y	16,387	(NULL)
ASIQ_IDX_T740_C2_FP	N	16,387	(NULL)

Index Information in System Views

Information on indexes is in the consolidated view `SYSINDEXES` and the system view `SYSIQIDX`.

See *Reference: Building Blocks, Tables, and Procedures > System Tables and Views* for a description of these views.

Removing Indexes

Remove column indexes that are no longer required.
Use either of the following methods:

- Enter a **DROP** statement.
  Use the SQL syntax in *Reference: Statements and Options > SQL Statements > DROP statement*.
- Delete the Index in SAP Sybase IQ.

Retaining Indexes When Removing Foreign Key Constraints

You may want to remove a foreign key constraint, but retain the underlying **HG** index. A non-unique **HG** index can provide query performance improvement, but may be expensive to build.
Enter an **ALTER TABLE DROP FOREIGN KEY** statement.

See the **ALTER INDEX** statement in *Reference: Statements and Options*.

Note: ALTER TABLE DROP FOREIGN KEY CONSTRAINT does not remove the automatically created nonunique **HG** index. You cannot drop a primary key if associated foreign keys remain. To remove such an index, drop it explicitly after issuing **ALTER TABLE DROP FOREIGN KEY**.

Optimizing Performance for Joins

You can create indexes to optimize join performance.

To gain the fastest processing of joins, create a Low_Fast (**LF**) or High_Group (**HG**) index on all columns that may be referenced in:

- **WHERE** clauses of join queries
- **HAVING** clause conditions of join queries outside of aggregate functions

For example:

```
SELECT n_name, sum(l_extendedprice*(1-l_discount))
    AS revenue
    FROM customer, orders, lineitem, supplier,
        nation, region
    WHERE c_custkey        = o_custkey
        AND o_orderkey     = l_orderkey
        AND l_suppkey      = s_suppkey
        AND c_nationkey    = s_nationkey
        AND s_nationkey    = n_nationkey
        AND n_regionkey    = r_regionkey
        AND r_name         = 'ASIA'
        AND o_orderdate   >= '1994-01-01'
        AND o_orderdate    < '1995-01-01'
GROUP BY n_name
HAVING n_name LIKE "I%"
    AND SUM(l_extendedprice*(1-l_discount)) > 0.50
ORDER BY 2 DESC
```

All columns referenced in this query except `l_extendedprice` and `l_discount` should have an **LF** or **HG** index.

Enforce Data Integrity

Table and column constraints and appropriate data type selection ensure that the data in your database is valid and reliable.

Data Integrity Overview

For data to have integrity means that the data is valid—that is, correct and accurate—and that the relational structure of the database is intact.

The relational structure of the database is described through *referential integrity* constraints, which are business rules that maintain the consistency of data between tables.

SAP Sybase IQ supports stored procedures and JDBC, which allow you detailed control over how data gets entered into the database. For information on JDBC, see the SAP Sybase IQ *Programming* guide.

How Data Can Become Invalid

Data may become invalid without proper checks.

For example:

- An operator enters orders to an orders table for a **customer_id** that does not exist in the customer table.
- An operator enters text where numeric data is required.
- An operator enters numeric data that is too wide for the column.
- More than one person enters the same information about a new department, with **dept_id** 200, to the department table.

Rules and Checks for Valid Data

To help ensure that the data in a database is valid, formulate checks that define valid and invalid data, and design the rules to which data must adhere.

Such rules are often called business rules. The collective name for checks and rules is constraints. Rules that maintain data integrity for a given column are column constraints. Rules that maintain integrity for one or more columns for a given table are table constraints. You can apply both table and column constraints to a single column in a table. Table constraints can also set the rule for a set of columns in a table.

Constraints that are built into the database itself are inherently more reliable than those built into client applications, or those spelled out as instructions to database users. Constraints built into the database are part of the database definition and can be enforced consistently across all applications.

Setting a constraint once, in the database, imposes it for all subsequent interactions with the database. By contrast, constraints that are built into client applications are vulnerable every time the software is altered, and may need to be imposed in several applications, or in several places in a single client application.

Declare any constraints that apply, whether or not SAP Sybase IQ enforces them. Declaring constraints ensures that you understand your data requirements, and are designing a database that matches the business rules of your organization.

SAP Sybase IQ performs several types of optimization based on the constraints you specify. This optimization does not depend on enforcement of constraints. For the best performance of queries and load operations, put all constraints in the database. **FOREIGN KEY**, **PRIMARY KEY** , **UNIQUE**, and **IQ UNIQUE** column constraints can improve load and query performance.

SAP Sybase IQ checks during load operations that certain constraints are obeyed. For example, SAP Sybase IQ ensures that data being loaded is the appropriate data type and length.

Statements That Change Database Contents

Client applications change information in database tables by submitting SQL statements.

Only a few SQL statements actually modify the information in a database:

- To delete an existing row of a table, use the **DELETE** statement.
- To insert a new row into a table, use the **INSERT** or **LOAD TABLE** statement.
- To change the value in a cell, use the **UPDATE** statement.

Data Integrity Tools

Use data constraints and constraints that specify the referential structure of the database to maintain data integrity.

Constraints

There are constraint types you can use on the data in individual columns or tables:

- A NOT NULL constraint prevents a column from containing a null entry. SAP Sybase IQ enforces this constraint.
- Columns can have CHECK conditions assigned to them, to specify that a particular condition should be met by every row for that column, for example, that salary column entries must be within a specified range.
- CHECK conditions can be made on the relative values in different columns, to specify, for example, in a library database that a `date_returned` entry is later than a `date_borrowed` entry.

Column constraints can be inherited from user-defined data types.

Entity and Referential Integrity

The information in relational database tables is tied together by the relations between tables. These relations are defined by the candidate keys and foreign keys that are built into the database design.

A *foreign key* is made up of a column or a combination of columns. Each foreign key relates the information in one table (the foreign table) to information in another (referenced or primary) table. A particular column, or combination of columns, in a foreign table is designated as a foreign key to the primary table.

The *primary key* or column (or set of columns) with a unique constraint is known as a *candidate key*. The referenced column or set of columns must be a candidate key and is called the *referenced key*. You cannot specify a foreign key constraint to a candidate key that is also a foreign key.

These referential integrity rules define the structure of the database:

- Keep track of the primary keys, guaranteeing that every row of a given table can be uniquely identified by a primary key that guarantees no nulls.
- Keep track of the foreign keys that define the relationships between tables. All foreign key values must either match a value in the corresponding primary key, if they are defined to allow NULL, or contain the NULL value.

Statements that Implement Integrity Constraints

Implement integrity constraints by submitting SQL statements.

- The **CREATE DATABASE** statement implements integrity constraints as the database is being created.
- The **ALTER DATABASE** statement adds integrity constraints, or deletes constraints, from an existing database.

Column Defaults Encourage Data Integrity

Column defaults automatically assign a specific value to a particular column or set of columns whenever a new row is entered into a database table.

A column default requires no action on the part of the client application. However, if the client application does specify a value for the column, the new value overrides the column default value.

Column defaults can quickly and automatically fill columns with information, such as the date or time a row is inserted or the user ID of the person who first modified a row in a table. Using column defaults encourages data integrity, but does not enforce it. Client applications can always override defaults.

Supported Default Values

SAP Sybase IQ supports default values for columns.

- A string specified in the **CREATE TABLE** statement or **ALTER TABLE** statement
- A number specified in the **CREATE TABLE** statement or **ALTER TABLE** statement
- An automatically incremented number: one more than the previous highest value in the column
- UUID (Universally Unique IDentifier) values generated by the **NEWID** function
- The current date, time, or timestamp
- The name of the current database
- The current user ID of the database user and the name of the user who last modified the row
- The publisher user ID of the database for SQL Remote applications
- A NULL value
- A constant expression, as long as it does not reference database objects
- A supported default value specified in a user-defined domain (data type) using the **CREATE DOMAIN** statement

Default Value Restrictions

Some column default values are not supported.

SAP Sybase IQ does not support the following values for column defaults:

- Values that use the special values **UTC TIMESTAMP**, **CURRENT UTC TIMESTAMP**, and **GLOBAL AUTOINCREMENT**
- A default value that is not compatible with the data type of the column
- A default value that violates the check constraint of the table or column
- A constant expression that references database objects

SAP Sybase IQ ignores settings for the `DEFAULT_TIMESTAMP_INCREMENT` database option.

Creating Column Defaults

You can use the **CREATE TABLE** statement to create column defaults when a table is created, or the **ALTER TABLE** statement to add column defaults later.

You can also specify a default value when creating a user-defined domain (data type) using the **CREATE DOMAIN** statement.

The stored procedure **sp_iqcolumn** returns information abut all columns for all tables. One of the column returned by the result set of **sp_iqcolumn** is called “default,” and shows the particular default value for that column.

1. Create a table named `tab1` with the default special value **LAST USER** specified for the `CHARACTER` column `c1`:

   ```
   CREATE TABLE tab1(c1 CHAR(20) DEFAULT LAST USER)
   ```

2. Add a condition to an existing column named `id` in the `sales_order` table, so that the value of the column automatically increments (unless a client application specifies a value):

   ```
   ALTER TABLE sales_order MODIFY id DEFAULT AUTOINCREMENT
   ```

3. Define a domain named `dom1`with a data type of `INTEGER` and a default value of 45:

   ```
   CREATE DOMAIN dom1 INTEGER DEFAULT 45
   ```

Changing Column Defaults

To change column defaults, use the same form of the **ALTER TABLE** statement you use to create defaults.
For example, to change the default value of a column named **order_date** from its current setting to **CURRENT DATE**, use:

```
ALTER TABLE sales_order
MODIFY order_date DEFAULT CURRENT DATE
```

Deleting Column Defaults

To remove column defaults, modify them to NULL.
This statement removes the default from the **order_date** column:

```
ALTER TABLE sales_order
MODIFY order_date DEFAULT NULL
```

Supported Column Default Values

You can load and insert column default values in SAP Sybase IQ.

Use these statements:

- **INSERT...VALUES**
- **INSERT...SELECT**
- **INSERT...LOCATION**
- **LOAD TABLE**
- **UPDATE**
- **SELECT...FROM...FOR UPDATE**

When you are defining and inserting column default values:

- You can specify default values that cannot be evaluated by SAP Sybase IQ. An error is reported when an **INSERT**, **LOAD**, or **ALTER ADD** operation is performed on a table that has an unsupported default value.
- SAP Sybase IQ generates an error or warning when the server attempts to insert a default value that is incompatible with the data type of the column. For example, if you define a

default expression of 'N/A' to an integer column, then any insert or load that does not specify the column value generates an error or warning, depending on the setting of the CONVERSION_ERROR database option.

- If a default value is too long for a CHARACTER type column, SAP Sybase IQ either truncates the string or generates an exception, depending on the setting of the STRING_RTRUNCATION database option.
- If the default value for a VARCHAR or LONG VARCHAR column is the zero-length string, SAP Sybase IQ inserts either a NULL or zero-length string, depending on the setting of the NON_ANSI_NULL_VARCHAR database option.
- If the default value for a VARCHAR, CHAR, or LONG VARCHAR column contains a partial multibyte character, SAP Sybase IQ may trim that character before inserting the value, depending on the setting of the TRIM_PARTIAL_MBC database option.
- If the default value violates the check constraint of either the table or the column, SAP Sybase IQ generates an error message every time the server attempts to insert the default value.
- Constraint violations that occur during a **LOAD TABLE** operation as a result of inserting default values are never ignored, regardless of any user-specified **IGNORE CONSTRAINT** and **MESSAGE LOG/ROW LOG** option.
- Column default values of **UTC TIMESTAMP** and **CURRENT UTC TIMESTMAP** are not supported by SAP Sybase IQ.
- Column default values defined on base tables are not propagated to the joins in which these tables participate.
- If a column on which a default value is defined is added to a table, all rows of the new column are populated with that default value.
- Changing the default value of an existing column in a table does not change any existing values in the table.
- The **LOAD TABLE DEFAULTS** option must be On to use the default value specified in the **LOAD TABLE** statement **DEFAULT** option. 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 DEFAULT** specification must contain at least one column to be loaded from the file specified in the **LOAD TABLE** command.
- The **LOAD TABLE DEFAULT** *default-value* must be of the same character set as that of the database, and 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.
- Encryption of the default value is not supported for the load default values specified in the **LOAD TABLE DEFAULT** clause.

Date, Time, and Timestamp Defaults

For columns with the DATE, TIME, or TIMESTAMP data type, you can use the **CURRENT DATE**, **CURRENT TIME**, **TIMESTAMP**, or **CURRENT TIMESTAMP** special value as a default. The default you choose must be compatible with the data type of the column.

CURRENT DATE Default

A **CURRENT DATE** default might be useful to record:

- Dates of phone calls in a contact database
- Dates of orders in a sales entry database
- The date a patron borrows a book in a library database

CURRENT TIMESTAMP Default

The **CURRENT TIMESTAMP** is similar to the **CURRENT DATE** default, but offers greater accuracy. For example, a user of a contact management application may have several contacts with a single customer in one day; the **CURRENT TIMESTAMP** default is useful for distinguishing between these contacts.

Since **CURRENT TIMESTAMP** records a date and the time down to a precision of millionths of a second, you may also find **CURRENT TIMESTAMP** useful when the sequence of events is important in a database.

TIMESTAMP Default

When a column is declared with **DEFAULT TIMESTAMP**, a default value is provided for insert and load operations. The value is updated with the current date and time whenever the row is updated.

On **INSERT** and **LOAD**, **DEFAULT TIMESTAMP** has the same effect as **CURRENT TIMESTAMP**. On **UPDATE**, if a column with a default value of **TIMESTAMP** is not explicitly modified, the value of the column is changed to the current date and time.

SAP Sybase IQ does not support default values of **UTC TIMESTAMP** or **CURRENT UTC TIMESTAMP**, nor does SAP Sybase IQ support the database option DEFAULT_TIMESTAMP_INCREMENT. SAP Sybase IQ generates an error every time an attempt is made to insert or update the default value of a column of type **UTC TIMESTAMP** or **CURRENT UTC TIMESTAMP**.

USER Defaults

Assigning a **DEFAULT USER** to a column identifies the person making an entry in a database. This information may be required; for example, when salespeople are working on commission.

Building a user ID default into the primary key of a table is a useful technique for occasionally connected users, and helps prevent conflicts during information updates. These users can make a copy of the tables that are relevant to their work on a portable computer, make changes

while not connected to a multiuser database, and then apply the transaction log to the server when they return.

USER Default

The special values **USER** and **CURRENT USER** return a string that contains the user ID of the current connection and can be used as a default value in columns with character data types. On **UPDATE**, columns with a default value of **USER** or **CURRENT USER** are not changed.

LAST USER Default

The special value **LAST USER** returns the name of the user who last modified the row and can be used as a default value in columns with character data types. On **INSERT** and **LOAD**, this constant has the same effect as **CURRENT USER**. On **UPDATE**, if a column with a default value of **LAST USER** is not explicitly modified, it is changed to the name of the current user.

When combined with the **DEFAULT TIMESTAMP**, a default value of **LAST USER** can be used to record (in separate columns) both the user and the date and time a row was last changed.

The IDENTITY or AUTOINCREMENT Default

The **IDENTITY/AUTOINCREMENT** default is useful for numeric data fields where the value of the number itself may have no meaning.

The feature assigns each new row a value of one greater than the current highest value in the column. You can use **IDENTITY/AUTOINCREMENT** columns to record purchase order numbers, to identify customer service calls or other entries where an identifying number is required.

Autoincrement columns are typically primary key columns or columns that are constrained to hold unique values (see **CREATE TABLE** statement in *Reference: Statements and Options*). For example, autoincrement default is effective when the column is the first column of an index, because the server uses an index or key definition to find the highest value.

You can sometimes retrieve the most recent value inserted into an autoincrement column using the *@@identity* global variable.

SAP Sybase IQ does not support the special value **GLOBAL AUTOINCREMENT**.

The initial **IDENTITY/AUTOINCREMENT** value is 0 when the table is created, and works only with positive integers.

A column with the **AUTOINCREMENT** default is referred to in Transact-SQL applications as an **IDENTITY** column. SAP Sybase IQ supports both keywords.

The NEWID Default

Use UUIDs (Universally Unique IDentifiers), also known as GUIDs (Globally Unique IDentifiers) to uniquely identify rows in a table.

The values are generated such that a value produced on one computer does not match a value produced on another computer. UUIDs can therefore be used as keys in replication and synchronization environments.

See the **NEWID** function in *Reference: Building Blocks, Tables, and Procedures*.

The NULL Default

For columns that allow NULL values, specifying a NULL default is exactly the same as not specifying a default.

If the client inserting the row does not explicitly assign a value, the row automatically receives a NULL value.

You can use NULL defaults when information for some columns is optional or not always available.

For more information about the NULL value, see *NULL Value* in *Reference: Building Blocks, Tables, and Procedures*.

String and Number Defaults

You can specify a specific string or number as a default value, as long as the column holds a string or number data type.

You must ensure that the default specified can be converted to the data type of the column.

Default strings and numbers are useful when there is a typical entry for a given column. For example, if an organization has two offices, the headquarters in **city_1** and a small office in **city_2**, you may want to set a default entry for a location column to **city_1**, to make data entry easier.

Constant Expression Defaults

You can use a constant expression as a default value, as long as the expression does not reference database objects.

You can use functions such as **GETDATE** and **DATEADD** in a constant expression default value. If the default constant expression is not a function or simple value, you must enclose the expression in parentheses.

For example, constant expressions allow column defaults to contain entries such as the date fifteen days from today:

```
... DEFAULT ( DATEADD( DAY, 15, GETDATE() ) )
```

Table and Column Constraints

Constraints help to ensure that the data entered into a table is correct, and provide information to SAP Sybase IQ that boosts performance.

The **CREATE TABLE** statement and **ALTER TABLE** statement can specify many different attributes for a table. Along with the basic table structure (number, name and data type of columns, name and location of the table), you can specify other features that allow control over data integrity.

Warning! Altering or creating tables might adversely interfere with other users of the database. For large tables, **ALTER TABLE** or **CREATE TABLE** can be a time-consuming operation. **CREATE TABLE** processing delays execution of other processes until the statement completes. Although you can execute **ALTER TABLE** statements while other connections are active, you cannot execute them if any other connection is using the table to be altered. During **ALTER TABLE**, no other requests referencing the table being altered are allowed while the statement is being processed.

UNIQUE Constraints on Columns or Tables

The **UNIQUE** constraint specifies that one or more columns uniquely identify each row in the table.

UNIQUE is essentially the same as a **PRIMARY KEY** constraint, except that you can specify more than one **UNIQUE** constraint in a table. With both **UNIQUE** and **PRIMARY KEY**, columns cannot contain any **NULL** values.

Example 1

This example adds the column `ss_number` to the `employee` table, and ensures that each value in it is unique throughout the table:

```
ALTER TABLE employee
ADD ss_number char(11) UNIQUE
```

Example 2

In this example, three columns make a unique entry.

```
ALTER TABLE product
ADD UNIQUE (name, size, color)
```

IQ UNIQUE Constraints on Columns

`IQ UNIQUE` defines the expected cardinality of a column and determines whether the column loads as `Flat FP` or `NBit`.

IQ UNIQUE defines the expected cardinality of a column and determines whether the column loads as Flat FP or NBit. 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 and FP_NBIT_LOOKUP_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.

CHECK Conditions on Columns

Use a CHECK condition to specify a criterion for the values in a column.

The condition may specify rules that data must satisfy in order to be reasonable, or rules that reflect organization policies and procedures.

CHECK conditions on individual column values are useful when only a restricted range of values are valid for that column.

Example 1

The entry should match one of a limited number of values. To specify that a `city` column only contains one of a certain number of allowed cities (say, those cities where the organization has offices), you could use a constraint similar to:

```
ALTER TABLE office
MODIFY city
CHECK ( city IN ( 'city_1', 'city_2', 'city_3' ) )
```

By default, string comparisons are case insensitive unless the database is explicitly created as a case-sensitive database, using the **CASE RESPECT** option.

Example 2

A date or number falls in a particular range. You may want to require that the **start_date** column of an employee table must be between the date the organization was formed and the current date, as in:

```
ALTER TABLE employee
MODIFY start_date
CHECK ( start_date BETWEEN '1983/06/27'
                   AND CURRENT DATE )
```

You can use several date formats: the YYYY/MM/DD format is always recognized regardless of the current option settings.

CHECK Conditions on User-Defined Data Types

You can attach CHECK conditions to user-defined data types. Columns defined on those data types inherit the CHECK conditions. A CHECK condition explicitly specified for a column overrides the CHECK condition from a user-defined data type.

When defining a CHECK condition on a user-defined data type, any variable prefixed with the @ sign is replaced by the name of the column when the CHECK condition is evaluated. For example, the following user-defined data type accepts only positive integers:

```
CREATE DATATYPE posint INT
CHECK ( @col > 0 )
```

You can use any variable name prefixed with @ instead of **@col**. Any column defined using the **posint** data type accepts only positive integers unless it has a different explicitly specified CHECK condition.

An **ALTER TABLE** statement with the DELETE CHECK clause deletes all CHECK conditions from the table definition, including those inherited from user-defined data types.

For information on user-defined data types, see *Reference: Building Blocks, Tables, and Procedures*.

CHECK Conditions on Columns

You can apply a CHECK condition as a constraint on a table, instead of on a single column.

Such CHECK conditions typically specify that two values in a row being entered or modified have a proper relation to each other. Column CHECK conditions are held individually in the system tables, and can be replaced or deleted individually. CHECK conditions on individual columns are recommended where possible.

Adding a Check Condition

You can add a new CHECK condition to a table or to an individual column.
For example, in a library database, the `date_returned` column for a particular entry must be the same as, or later than the `date_borrowed` entry:

```
ALTER TABLE loan
ADD CHECK(date_returned >= date_borrowed)
```

Deleting a Check Condition

You can delete a CHECK condition from an individual column.

To delete a CHECK condition on a column, set it to NULL.

This statement removes the CHECK condition on the **phone** column in the **customer** table:

```
ALTER TABLE customer MODIFY phone
CHECK NULL
```

Replacing a Check Condition

You can delete a CHECK condition from an individual column.

You can replace a CHECK condition on a column in the same way as you would add a CHECK condition.

This statement adds or replaces a CHECK condition on the `city` column of the `office` table:

```
ALTER TABLE office
MODIFY city
CHECK ( city IN ( 'city_1', 'city_2', 'city_3' ) )
```

CHECK Conditions on Tables

There are two ways to modify a CHECK condition defined on a table.

- Add a new CHECK condition using **ALTER TABLE** with an ADD table-constraint clause.
- Delete all existing CHECK conditions, including column CHECK conditions, using **ALTER TABLE** DELETE CHECK, and then add in new CHECK conditions.

Deleting a column from a table does not delete CHECK conditions associated with the column that are held in the table constraint. If the constraints are not removed, any attempt to query data in the table produces a `column not found` error message.

Removing Check Conditions from Tables

Use the **ALTER TABLE** statement with the **DELETE CHECK** clause to remove all CHECK conditions on a table, including CHECK conditions on all its columns and CHECK conditions inherited from user-defined data types.
For example:

```
ALTER TABLE table_name
DELETE CHECK
```

Entity and Referential Integrity

Once you specify the primary key for each table, no further action is needed by client application developers or the database administrator to maintain entity integrity.

The table owner defines the primary key for a table when creating it. If the structure of a table is later modified, the primary key can also be redefined using the **ALTER TABLE** statement clauses **DELETE PRIMARY KEY** or **ADD PRIMARY KEY**. See *Reference: Statements and Options*.

Some application development systems and database design tools allow you to create and alter database tables. If you are using such a system, you may not have to explicitly enter the **CREATE TABLE** or **ALTER TABLE** command. The application generates the statement from the information you provide.

When you insert or update a table row, the database server ensures that the primary key for the table is still valid: that each row in the table is uniquely identified by the primary key.

Example 1

The **Employees** table in the demo database uses an employee ID as the primary key. When a new employee is added to the table, IQ checks that the new employee ID value is unique, and is not NULL. See *Table Names and Owners* in *Introduction to SAP Sybase IQ* for a list of tables in the demo database.

Example 2

The **SalesOrderItems** table in the demo database uses two columns to define a primary key.

This table holds information about the items that can be ordered. One column contains an **id** that specifies an order, but there may be several items on each order, so this column by itself cannot be a primary key. An additional **line_id** column identifies the line that corresponds to the item. Together, the columns **id** and **line_id** specify an item uniquely, and form the primary key. This is known as a *multicolumn primary key*.

If a Client Application Breaches Entity Integrity

Entity integrity requires that each value of a primary key or unique constraint be unique within the table, and that there are no NULL values.

If a client application attempts to insert or update a primary key value, and provides values that are not unique, entity integrity is breached.

A breach in entity integrity prevents the new information from being added to the database, and instead sends the client application an error.

The application programmer should decide how to present this information to the user and enable the user to take appropriate action. The appropriate action in this case is usually just to provide a unique value for the primary key.

SAP Sybase IQ checks referential integrity for each **UPDATE** on a foreign key or candidate key, each **DELETE** on a candidate key, and each **LOAD/INSERT** on a foreign key. When a referential integrity violation occurs, **UPDATE** or **DELETE** requests are immediately denied and rolled back. **LOAD/INSERT** requests that violate referential integrity are also denied or rolled back. SAP Sybase IQ also optionally rejects rows that violate data integrity as specified by the user.

Referential Integrity

The entries in a foreign key must correspond to the primary key values of a row in the referenced table.

Occasionally, some other unique column combination may be referenced instead of a primary key. The primary key or column (or set of columns) with a unique constraint is known as a

candidate key. The referenced column or set of columns must be a candidate key and is called the **referenced key**.

Loss of Referential Integrity

SAP Sybase IQ provides protection against referential integrity loss.

Your database can lose referential integrity if someone:

- Updates or deletes a primary key value that has a matching foreign key value. All the foreign keys referencing that primary key violate referential integrity.
- Adds a new row to the foreign table, and enters a value for the foreign key that has no corresponding candidate key value. The database violates referential integrity.

SAP Sybase IQ provides protection against both types of integrity loss.

When a referenced candidate key is updated or deleted, SAP Sybase IQ disallows **UPDATE** or **DELETE**.

Foreign Keys

Use the **CREATE TABLE** statement or **ALTER TABLE** statement to create foreign keys, as you do primary keys.

Note: You cannot create foreign key constraints on local temporary tables. Global temporary tables must be created with **ON COMMIT PRESERVE ROWS.**

The demo database contains an employee table and a department table. The primary key for the employee table is the employee ID, and the primary key for the department table is the department ID.

For example, assume the following schema:

```
DEPT table
{ DeptNo int primary key
DeptName varchar(20),
Mgr int,
foreign key MGR_EMPNO (Mgr) references EMPLOYEE(EmpNo) on update
restrict }

EMPLOYEE table
{ EmpNo int primary key,
DeptNo int references DEPT(DeptNo) on delete restrict,
LastName varchar(20),
FirstName varchar(20),
Salary int }
```

In the employee table, the department ID is a foreign key for the department table; each department ID in the employee table corresponds exactly to a department ID in the department table.

The foreign key relationship is a many-to-one relationship, and, in this example, is mandatory. Several entries in the employee table have the same department ID entry, but the department ID is the primary key for the department table, and so is unique. If a foreign key was allowed to

reference a column with duplicate entries in the department table, there is no way to determine which row in the department table is the appropriate reference. This is a mandatory foreign key.

Referential Integrity Violations

SAP Sybase IQ supports referential integrity with RESTRICT action (the ANSI default) at the statement level.

This means that SAP Sybase IQ denies requests for updates and deletes on a primary key or any column with a unique constraint that removes any value upon which correspondent foreign keys depend. (You must carefully consider the order in which you request deletes and updates.) SAP Sybase IQ issues an error message and rolls back load operations that violate referential integrity, but does allow you to specify that certain rows can be ignored.

Enforcing Referential Integrity with Existing Unenforced Foreign Keys

You can enforce referential integrity with unenforced foreign keys.

Assume the following schema:

```
DEPT table
{ DeptNo int primary key
DeptName varchar(20),
Mgr int,
foreign key MGR_EMPNO (Mgr) references EMPLOYEE(EmpNo) on update
restrict }

EMPLOYEE table
{ EmpNo int primary key,
DeptNo int references DEPT(DeptNo) on delete restrict,
LastName varchar(20),
FirstName varchar(20),
Salary int }
```

1. Identify the candidate key to foreign key relationship.

   In the schema, there are two such relationships:
   - Foreign key(EMPLOYEE.DeptNo to Candidate key(DEPT.DeptNo)
   - Foreign key(DEPT.Mgr) to Candidate key (EMPLOYEE.EMPNo)
2. Add a primary key or unique constraint on the candidate key via the **ALTER TABLE** statement if none exist. (In the preceding example, the primary key already exists.) All candidate key values must be unique and non-null.
3. Drop any existing unenforced foreign key constraint, for example:

   ```
   ALTER TABLE DEPT DROP FOREIGN KEY MGR_EMPNO;
   ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPT;
   ```

   In the schema, the unenforced foreign key constraints MGR_EMPNO and EMPLOYEE(DeptNo) referencing DEPT(DeptNo) must be dropped. If there is no user

specified role name for EMPLOYEE(DeptNo) to DEPT(DeptNo), the default role name is the same as the primary table, in other words, DEPT.

4. Add the foreign key constraint(s). For example:

```
ALTER TABLE DEPT ADD FOREIGN KEY MGR_EMPNO(Mgr) REFERENCES
EMPLOYEE(EmpNo);
ALTER TABLE EMPLOYEE ADD FOREIGN KEY EMP_DEPT(DeptNo) REFERENCES
DEPT(DeptNo);
```

Enforcing Referential Activity in a New Table

Enforce referential integrity in a new table.

1. Create the primary table, for example:

```
CREATE TABLE DEPT(DeptNo int primary key,
DeptName varchar(20),
Mgr int );
```

2. Create the foreign table. For example, in this statement, the default role name for the specified foreign key is DEPT:

```
CREATE TABLE EMPLOYEE(EmpNo int primary key,
DeptNo int references DEPT(DeptNo)
on delete restrict,
LastName varchar(20),
FirstName varchar(20),
Salary int);
```

3. Add the foreign key constraint. For example:

```
CREATE TABLE EMPLOYEE(EmpNo int primary key,
DeptNo int,
LastName varchar(20),
FirstName varchar(20),
Salary int,
FOREIGN KEY EMP_DEPT(DeptNo) REFERENCES DEPT(DeptNo));
```

In this statement, the user specified role name for the same foreign key is EMP_DEPT:

```
CREATE TABLE EMPLOYEE(EmpNo int primary key,
DeptNo int,
LastName varchar(20),
FirstName varchar(20),
Salary int,
FOREIGN KEY EMP_DEPT(DeptNo) REFERENCES DEPT(DeptNo));
```

Dropping a Foreign Key

Drop a foreogn key from a table.

1. Create the primary table and foreign table. For example:

```
CREATE TABLE DEPT(DeptNo int primary key,
DeptName varchar(20),
Mgr int );
```

```
CREATE TABLE EMPLOYEE(EmpNo int primary key,
DeptNo int references DEPT(DeptNo)
on delete restrict,
```

```
LastName varchar(20),
FirstName varchar(20),
Salary int);
```

2. When there is no role name assigned, the default role name for the specified foreign key is DEPT:

```
ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPT;
```

3. If there are multiple foreign keys and the role name is unknown, you can use the **sp_iqconstraint** procedure to display it. See *Reference: Building Blocks, Tables, and Procedures*.
4. If a role name was assigned, EMP_DEPT, for example, you must specify it when dropping the key:

```
ALTER TABLE EMPLOYEE DROP FOREIGN KEY EMP_DEPT;
```

These statements do not drop the non-unique HG index for EMPLOYEE(DeptNo) which is implicitly created. To drop it, use **sp_iqindex** to find the HighGroup index name and use the **DROP INDEX** statement, as follows:

```
sp_iqindex('EMPLOYEE');
```

```
EMPLOYEE DBA DeptNO FP ASIQ_IDX_T27_C2_FP N
EMPLOYEE DBA DeptNO HG ASIQ_IDX_T27_C2_HG N
EMPLOYEE DBA EmpNO FP ASIQ_IDX_T27_C1_FP N
EMPLOYEE DBA EmpNO HG ASIQ_IDX_T27_I11_HG N
EMPLOYEE DBA FirstName FP ASIQ_IDX_T27_C4_FP N
EMPLOYEE DBA LastName FP ASIQ_IDX_T27_C3_FP N
EMPLOYEE DBA Salary FP ASIQ_IDX_T27_C5_FP N
```

```
DROP INDEX ASIQ_IDX_T27_C2_HG
```

To drop a table, you must drop all associated foreign key constraints. Drop foreign key constraint and tables in this order:

```
ALTER TABLE DROP FOREIGN KEY MGR_EMPNO;
DROP TABLE EMPLOYEE;
DROP TABLE DEPT;
```

Another way to drop the same tables would be to use the following two **ALTER TABLE** statements in any order and then do **DROP TABLE** statements in any order:

```
ALTER TABLE DEPT DROP FOREIGN KEY MGR_EMPNO;
ALTER TABLE EMPLOYEE DROP FOREIGN KEY EMP_DEPT;
```

Suppose that the database also contained an office table, listing office locations. The employee table might have a foreign key for the office table that indicates where the employee's office is located. The database designer may allow for an office location not being assigned when the employee is hired. In this case, the foreign key should allow the NULL value for when the office location is unknown or when the employee does not work out of an office.

Concurrent Operations

The referential integrity feature of SAP Sybase IQ restricts concurrent updates or deletes on a primary table during loads or inserts on a foreign table.

Table 19. Concurrent Operations That Return an Error

First request	Request of Overlapping Transaction
Request by one transaction for **LOAD/INSERT/UPDATE/ ALTER TABLE ADD** foreign key/ **ALTER TABLE DROP** foreign key to any foreign table	**DELETE** its associated primary table with deletable row(s).
	UPDATE its associated primary table.
	TRUNCATE its associated primary table.

SAP Sybase IQ also generates an error for a request by one transaction to alter a table to add a foreign key or drop a foreign key while there are old versions of the foreign table, the primary table, or both, in use by other transactions.

For both enforced and unenforced foreign key and primary key, SAP Sybase IQ allows:

- Simultaneous load or insert on one or more foreign tables and the shared primary table.
- Simultaneous load or insert on foreign tables and delete, or update, or truncate on another one or more foreign tables.
- Simultaneous delete, or update, or truncate on two or more foreign tables, even if sharing the same primary table.
- Simultaneous delete, or update, or truncate on foreign tables and delete, or update, or truncate on the shared primary table.
- **ALTER TABLE ADD** foreign key or **DROP** foreign key if no transaction is using any old version(s) of foreign/primary table and these unused old version(s) will be dropped as part of the **ADD/DROP** foreign key operation.

Concurrent Operations on Foreign and Primary Tables

The table-level versioning of SAP Sybase IQ guarantees consistent referential integrity checks while allowing concurrent load, or insert, or update operations on the foreign table and **LOAD/INSERT** operations on the primary table.

SAP Sybase IQ also verifies that there are no deleted old values in a foreign table when a transaction requesting **DELETE** or **UPDATE** starts. This provides consistent referential integrity checking during a concurrent delete operation on a foreign table and a delete or update operation on a PRIMARY Table.

Assume that there are two foreign key constraints among two foreign tables, `ftab1` and `ftab2`, and one primary table, `ptab`. Assume that foreign key `ftab1(fk1,fk2)` references candidate key `ptab(pk1,pk2)`. Foreign key `ftab2(fk1,fk2)` references the same candidate key. Candidate key `ptab(pk1,pk2)` can either be a primary key or a unique constraint.

This table shows which operations on both tables (foreign and primary) should be allowed and which return an error. Information in the table applies only to enforced foreign keys and candidate key.

Table 20. Concurrent DML on Foreign and Primary Tables

	LOAD or INSERT ftab1	DELETE/ TRUNCATE TABLE ftab1	UPDATE ftab1 (fk1,fk2)	Populate new index non-FK ftab1 (fk1,fk2)	ADD FK ftab1 (fk1 fk2)	DROP FK ftab1 (fk2, fk2)
LOAD ftab2	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
LOAD ptab	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
INSERT ftab2	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
INSERT ptab	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
DELETE ftab2 TRUNCATE TABLE ftab2	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
DELETE ptab TRUNCATE TABLE ptab	Error	Allowed	Error	Allowed	Error	Error
UPDATE ftab2(fk1,fk2)	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
UPDATE ptab (pk1,pk2)	Error	Allowed	Error	Allowed	Error	Error
Populate new index	Allowed	Allowed	Allowed	Allowed	Allowed	Allowed
QUERY (old version of ftab1/ptab in use with or without (fk1,fk2))	Allowed	Allowed	Allowed	Allowed	Error	Error
No old version of ftab2 in use	N/A	N/A	N/A	N/A	Allowed (drop all unused old versions of ftab1)	Allowed (drop all unused old versions of ftab1)

Concurrency conflict occurs if one transaction loads foreign key columns while another updates associated candidate key columns. There is no conflict if one transaction loads foreign

key columns while another updates unassociated candidate key columns on one of its associated candidate tables.

Note: For efficient performance, a query on union all views opens the tables referred to by those columns used as join keys or group by columns. Until the transaction commits and read locks on the tables are released, you cannot alter or drop the tables that have foreign keys used as join conditions or grouping columns. You can, however, load, insert, delete, and update these tables while the query is running.

Disabling Referential Integrity Checking

You can use the SAP Sybase IQ option `DISABLE_RI_CHECK` to bypass referential integrity checking.

Because bypassing referential integrity checking defeats the purpose of having the feature, use this option carefully.

Integrity Rules in System Tables

All the information about integrity checks and rules in a database is held in system tables and views.

System Table	Description
`SYS.SYSTABLE`	CHECK constraints are held in the `view_def` column of `SYS.SYSTABLE`. For views, the `view_def` holds the **CREATE VIEW** command that created the view. You can check whether a particular table is a base table or a view by looking at the `table_type` column, which is BASE or VIEW.
`SYS.SYSFOREIGNKEYS`	This view presents the foreign key information from the two tables `SYS.SYSFOREIGNKEY` and `SYS.SYSFKCOL` in a more readable format.
`SYS.SYSCOLUMNS`	This view presents the information from the `SYS.SYSCOLUMN` table in a more readable format. It includes default settings and primary key information for columns.

You can use Interactive SQL to browse these tables and views.

Validating Catalog Store Indexes

You can validate an index on SQL Anywhere tables in the catalog store to ensure that every row referenced in the index actually works in the table.
To validate an index, open a command prompt and run the **dbvalid** utility.

For example, the following statement, which should be typed on a single line) validates an index called EmployeeIndex. The **-i** switch specifies that each object name given is an index.

```
dbvalid -c "uid=dba;pwd=sql;eng=myserver"
-i EmployeeIndex
```

Next

See *dbvalid Database Administration Utility* in the Utility Guide.

Manage Transactions and Versioning

Transaction processing ensures that logically related groups commands execute as a unit. SAP Sybase IQ uses transaction processing to allow many users to read from the database while it is being updated. Transactions are fundamental to maintaining the accuracy of your data, and to data recovery in the event of system failure.

A crucial aspect of transaction processing is its ability to isolate users from the effect of other users' transactions. The SAP Sybase IQ approach to transaction processing, called *snapshot versioning*, supports the highest level of isolation recognized by ISO. Snapshot versioning provides each database user with a snapshot of the database. Any changes that you make to the database are not seen by other database users until you commit your transaction.

See also

- *Table-Level Snapshot Versioning* on page 234
- *Row-Level Snapshot Versioning* on page 239

Transactions

Each transaction is a sequence of logically related commands that accomplish one task and transform the database from one consistent state into another.

Transactions are atomic; SAP Sybase IQ executes all the statements within a transaction as a unit. At the end of each transaction, changes are committed to make them permanent. If for any reason any of the commands in the transaction do not process properly, some or all of the intermediate changes can be undone, or *rolled back*. The user application controls the conditions under which changes are committed or rolled back.

Transactions are made up of small blocks. The completion of each block marks a point at which the information is self-consistent. Transaction processing is fundamental to ensuring that a database contains correct information.

In most cases, SAP Sybase IQ transactions begin and end automatically, based on the commands being issued, and the options set. You can also issue explicit commands to begin or end a transaction.

Transactions start automatically with the first statement following either:

- A connection to a database
- The end of a previous transaction

SAP Sybase IQ also supports SAP Sybase IQ Transact-SQL commands, such as **BEGIN TRANSACTION**, for compatibility with Adaptive Server Enterprise. You can explicitly start a transaction using the **BEGIN TRANSACTION** command.

Transactions complete with one of these events:

- A **COMMIT** statement makes the changes to the database permanent.
- A **ROLLBACK** statement undoes all the changes made by the transaction.
- A disconnection from a database causes an implicit rollback (the default) or commit.
- The execution of a statement with a side effect of an *automatic commit* makes changes to the database.

Database definition commands, such as **ALTER**, **CREATE**, and **DROP** all have the side effect of an automatic commit.

You can use savepoints to identify important states within a transaction and return to them selectively or cause other actions to occur.

See also

- *CHECKPOINT Statement* on page 440
- *SAVEPOINT Statement* on page 495

Viewing Transaction Activity

Use the **sp_iqtransaction** stored procedure to display a snapshot of transaction activity, such as main and temporary space created and in use, open cursors, and savepoints.
Run **sp_iqtransaction**.

The procedure returns a row for each transaction control block in the IQ transaction manager.

sp_iqtransaction Procedure

Shows information about transactions and versions.

Syntax

```
sp_iqtransaction
```

Applies to

Simplex and multiplex.

Privileges

Requires the MONITOR system privilege. Users without the MONITOR system privilege must be granted `EXECUTE` permission to run the stored procedure.

Description

sp_iqtransaction returns a row for each transaction control block in the SAP Sybase IQ transaction manager. The columns Name, Userid, and ConnHandle are the connection properties **Name**, **Userid**, and **Number**, respectively. Rows are ordered by TxnID.

sp_iqtransaction output does not include connections without transactions in progress. To include all connections, use **sp_iqconnection**.

Note: Although you can use **sp_iqtransaction** to identify users who are blocking other users from writing to a table, **sp_iqlocks** is a better choice for this purpose.

Column Name	Description
Name	The name of the application.
Userid	The user ID for the connection.
TxnID	The transaction ID of this transaction control block. The transaction ID is assigned during **begin transaction**. It appears in the `.iqmsg` file by the BeginTxn, CmtTxn, and PostCmtTxn messages, and is the same as the Txn ID Seq that is logged when the database is opened.
CmtID	The ID assigned by the transaction manager when the transaction commits. For active transactions, the CmtID is zero.
VersionID	For simplex and multiplex nodes, a value of 0 indicates that the transaction is unversioned, and the VersionID has not been assigned. For the multiplex coordinator, the VersionID is assigned after the transaction establishes table locks. Multiplex secondary servers receive the VersionID from the coordinator. The VersionID is used internally by the SAP Sybase IQ in-memory catalog and the IQ transaction manager to uniquely identify a database version to all nodes within a multiplex database.
State	The state of the transaction control block. This variable reflects internal SAP Sybase IQ implementation details and is subject to change in the future. Currently, transaction states are `NONE`, `ACTIVE`, `ROLLING_BACK`, `ROLLED_BACK`, `COMMITTING`, `COMMITTED`, and `APPLIED`. `NONE`, `ROLLING_BACK`, `ROLLED_BACK`, `COMMITTING` and `APPLIED` are transient states with a very small life span. `ACTIVE` indicates that the transaction is active. `COMMITTED` indicates that the transaction has completed and is waiting to be `APPLIED`, at which point a version that is invisible to any transaction is subject to garbage collection. Once the transaction state is `ROLLED_BACK`, `COMMITTED`, or `APPLIED`, ceases to own any locks other than those held by open cursors.
ConnHandle	The ID number of the connection.
IQConnID	The ten-digit connection ID that is included as part of all messages in the `.iqmsg` file. This is a monotonically increasing integer unique within a server session.

Column Name	Description
MainTableKBCr	The number of kilobytes of IQ store space created by this transaction.
MainTableKBDr	The number of kilobytes of IQ store space dropped by this transaction, but which persist on disk in the store because the space is visible in other database versions or other savepoints of this transaction.
TempTableKBCr	The number of kilobytes of IQ temporary store space created by this transaction for storage of IQ temporary table data.
TempTableKBDr	The number of kilobytes of IQ temporary table space dropped by this transaction, but which persist on disk in the IQ temporary store because the space is visible to IQ cursors or is owned by other savepoints of this transaction.
TempWorkSpaceKB	For ACTIVE transactions, a snapshot of the work space in use at this instant by this transaction, such as sorts, hashes, and temporary bitmaps. The number varies depending on when you run **sp_iqtransaction**. For example, the query engine might create 60MB in the temporary cache but release most of it quickly, even though query processing continues. If you run **sp_iqtransaction** after the query finishes, this column shows a much smaller number. When the transaction is no longer active, this column is zero. For ACTIVE transactions, this column is the same as the TempWorkSpaceKB column of **sp_iqconnection**.
TxnCreateTime	The time the transaction began. All SAP Sybase IQ transactions begin implicitly as soon as an active connection is established or when the previous transaction commits or rolls back.
CursorCount	The number of open SAP Sybase IQ cursors that reference this transaction control block. If the transaction is ACTIVE, it indicates the number of open cursors created within the transaction. If the transaction is COMMITTED, it indicates the number of hold cursors that reference a database version owned by this transaction control block.
SpCount	The number of savepoint structures that exist within the transaction control block. Savepoints may be created and released implicitly. Therefore, this number does not indicate the number of user-created savepoints within the transaction.
SpNumber	The active savepoint number of the transaction. This is an implementation detail and might not reflect a user-created savepoint.
MPXServerName	Indicates if an active transaction is from an internode communication (INC) connection. If from INC connection, the value is the name of the multiplex server where the transaction originates. NULL if not from an INC connection. Always NULL if the transaction is not active.

Column Name	Description
GlobalTxnID	The global transaction ID associated with the current transaction, 0 (zero) if none.
VersioningType	The snapshot versioning type of the transaction; either table-level (the default), or row-level. Row-level snapshot versioning (RLV) applies only to RLV-enabled tables. Once a transaction is started, this value cannot change.
Blocking	Indicates if connection blocking is enabled (True) or disabled (False). You set connection blocking using the **BLOCKING** database option. If true, the transaction blocks, meaning it waits for a conflicting lock to release before it attempts to retry the lock request.
BlockingTimeout	Indicates the time, in milliseconds, a transaction waits for a locking conflict to clear. You set the timeout threshold using the **BLOCKING_TIMEOUT** database option. A value of 0 (default) indicates that the transaction waits indefinitely.

Example

Example **sp_iqtransaction** output:

```
Name   Userid  TxnID  CmtID VersionID    State    ConnHandle  IQConnID
====== ====== ====== ====== =========  ========== ===========
========
red2       DBA  10058  10700     10058  Active     419740283       14


MainTableKBCr      MainTableKBDr    TempTableKBCr TempTableKBDr
============= ================== ================ =============
           0                  0            65824             0


TempWorkSpaceKB TxnCreateTime               CursorCount SpCount
SpNumber
==============
=======================     =========== ======= ========
       0        2013-03-26 13:17:27.612            1       3        2


MPXServerName  GlobalTxnID   VersioningType  Blocking
BlockingTimeout
=============  ===========   ==============  ========
===============
        (NULL)          0     Row-level       True
0
```

Isolation Levels

An important aspect of transaction processing is the database server's ability to isolate an operation. ANSI standards define four levels of isolation. Each higher level provides

transactions a greater degree of isolation from other transactions, and thus a greater assurance that the database remains internally consistent.

The isolation level controls the degree to which operations and data in one transaction are visible to operations in other, concurrent transactions. Table-level snapshot versioning supports the highest level of isolation. At this level, all schedules may be serialized.

Table-level snapshot versioning maintains this high level of isolation between concurrent transactions by following these rules:

- Transaction management maintains a snapshot of committed data when each transaction begins.
- A transaction can always read, as long as the snapshot version it uses is maintained.
- A transaction's writes are reflected in the snapshot it sees.
- Once a transaction begins, updates made by other transactions are invisible to it.

The level of isolation that table-level snapshot versioning provides prevents several types of inconsistencies, including (most commonly encountered):

- **Dirty reads –** transaction A modifies an object, but does not commit or roll back the change. Transaction B reads the modified object. Then transaction A further changes the object before performing a **COMMIT**. In this situation, transaction B has seen the object in a state that was never committed.
- **Non repeatable reads –** transaction A reads an object. Transaction B then modifies or deletes the object and performs a **COMMIT**. If transaction A attempts to read the same object again, it will have been changed or deleted.
- **Phantom data elements –** transaction A reads a set of data that satisfies some condition. Transaction B then executes an **INSERT** and then a **COMMIT**. The newly committed data now satisfies the condition, when it did not previously. Transaction A then repeats the initial read and obtains a different set of data.
- **Lost updates –** in an application that uses cursors, Transaction A writes a change for a set of data. Transaction B then saves an update that is based on earlier data. The changes of Transaction A are completely lost.

Table-level snapshot versioning ensures that only one user can modify a table at any given time, keeps the changes invisible to other users until the changes are complete, and maintains timestamped snapshots of data objects in use at any time.

While IQ allows you to set the isolation level to 0, 1, 2, or 3 (comparable to ANSI levels 1, 2, 3, or 4), there is no reason to do so. All users execute at isolation level 4, even if you set a different level. There is no performance advantage to setting a lower isolation level.

For more information on preventing concurrent transactions from accessing or modifying tables, see the **LOCK TABLE** statement in *Reference: Statements and Options*.

Transaction Blocking

When you set the **BLOCKING** option to on, any transaction attempting to obtain a write lock that conflicts with an existing write lock held by another transaction waits until the conflicting lock is released, or until the **BLOCKING_TIMEOUT** threshold is reached. By default, **BLOCKING** is off.

Connection blocking puts the requesting transaction to sleep (*blocking* the table-level or row-level lock) until the connection holding the lock releases it (*unblocking* the table-level or row-level lock). You control the duration of the block by setting the **BLOCKING_TIMEOUT** value in milliseconds. When **BLOCKING_TIMEOUT** is 0 (the default), all blocked transactions in the connection wait indefinitely until the connection obtains the requested lock. When **BLOCKING_TIMEOUT** is set, and the lock is not released within the specified time, then the waiting transaction receives an error message.

Blocking takes advantage of delayed transaction versioning, where the transaction manager creates the transaction snapshot version after establishing a table-level or row-level lock.

Consider this blocking example for a table-level lock:

- User A connects.
- User A executes `INSERT INTO iq_table1 VALUES( 3, 300);`.
- User B connects.
- User B executes `INSERT INTO iq_table1 VALUES( 4, 400);`.

With connection blocking disabled, User B's transaction is rolled back, and he or she receives an error indicating that another user has a lock on `iq_table1`.

With connection blocking enabled (and **BLOCKING_TIMEOUT** set to 0), User B's **INSERT** statement is put to sleep until User A's **INSERT** statement commits, releasing the table write lock.

With connection blocking enabled and **BLOCKING_TIMEOUT** set to 200 milliseconds, User B's transaction is rolled back and he or she receives an error indicating that another user has a lock on `iq_table1`, if User A's transaction does not commit within the 200 millisecond blocking timeout threshold.

For row-level lock examples, see *Administration: In-Memory Row-Level Versioning > Manage Blocking in RLV Store*.

Enabling Connection Blocking

Enable connection blocking to force any transaction attempting to obtain a lock that conflicts with another transaction's existing lock to wait: either until every conflicting lock is released, or until the **BLOCKING_TIMEOUT** threshold is reached.

Prerequisites

- SAP Sybase IQ server has a simplex database.

Task

Set the **BLOCKING** database option to ON.

```
set temporary option blocking = 'On';
```

Note: The blocking option can be set either at the connection or PUBLIC level.

Disabling Connection Blocking

Disable connection blocking to force any transaction attempting to obtain a lock that conflicts with another transaction's existing lock to roll back the transaction and display an error.

Prerequisites

- SAP Sybase IQ server has a simplex database.

Task

Set the **BLOCKING** database option to OFF.

```
set temporary option blocking = 'Off';
```

Note: The blocking option can be set either at the connection or PUBLIC level.

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.

Setting the Blocking Timeout Threshold

Use the threshold to set the length of time, in milliseconds, a transaction waits to obtain a lock. If the transaction attempting to obtain a lock conflicts with another transaction's existing lock, it waits until the **BLOCKING_TIMEOUT** option threshold is reached. If the conflict still exists, the transaction rolls back and you see an error.

Prerequisites

- SAP Sybase IQ server has a simplex database.

Task

Note: The default value, 0, indicates that a blocked transaction must wait indefinitely until all conflicting transactions release their locks.

Set the **BLOCKING_TIMEOUT** database option value to the number of milliseconds you want the transaction to wait for conflicting transactions to release their locks.

```
set temporary option blocking_timeout = '400';
```

Note: The blocking option can be set either at the connection or PUBLIC level.

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.

Transaction Blocking Deadlocks

Transaction blocking can lead to a deadlock situation, in which a set of transactions arrive at a state where none of them can proceed.

A deadlock can arise for two reasons:

- **Cyclical blocking conflict –** transaction A is blocked on transaction B, and transaction B is blocked on transaction A. Additional time cannot solve the problem, and one of the transactions must be canceled, allowing the other to proceed. The same situation can arise with more than two transactions blocked in a cycle.

  To eliminate a transactional deadlock, the database server selects a connection from those involved in the deadlock, rolls back the changes for the transaction that is active on that connection and returns an error. The database server selects the connection to roll back by using an internal heuristic that prefers the connection with the smallest blocking wait time left as determined by the **BLOCKING_TIMEOUT** option. If all connections are set to wait forever, then the connection that caused the server to detect a deadlock is selected as the victim connection.
- **All workers are blocked –** when a transaction becomes blocked, its worker is not relinquished. For example, a database server is configured with three workers. Transactions A, B, and C are blocked on transaction D, which is not currently executing a request. A deadlock situation arises because there are no available workers. This situation is called thread deadlock.

  Suppose that the database server has *n* workers. Thread deadlock occurs when *n*-1 workers are blocked, and the last worker is about to block. The database server's kernel cannot permit this last worker to block, since doing so results in all workers being blocked, and the

database server stops responding. Instead, the database server ends the task that is about to block the last worker, rolls back the changes for the transaction active on that connection, and returns an error.

Database servers with tens or hundreds of connections may experience thread deadlocks in long-running requests, either because of the size of the database or because of blocking. In this case, you may want to increase the value of the **-gn** server option of the **start_iq** utility.

To view locks and deadlocks in Sybase Control Center, see the Sybase Control Center online help.

Creating a Deadlock Reporting Event in Interactive SQL

Create a table and a system event for obtaining information about deadlocks.

Prerequisites

SAP Sybase IQ server has a simplex database.

Task

1. Create a table to store the data returned from the **sa_report_deadlocks** system procedure.

```
CREATE TABLE DeadlockDetails(
  deadlockId INT PRIMARY KEY DEFAULT AUTOINCREMENT,
  snapshotId BIGINT,
  snapshotAt TIMESTAMP,
  waiter INTEGER,
  who VARCHAR(128),
  what LONG VARCHAR,
  object_id UNSIGNED BIGINT,
  record_id BIGINT,
  owner INTEGER,
  is_victim BIT,
  rollback_operation_count UNSIGNED INTEGER );
```

2. Create an event that sends an e-mail notification when a deadlock occurs.

```
CREATE EVENT DeadlockNotification
TYPE Deadlock
HANDLER
BEGIN
 INSERT INTO DeadlockDetails WITH AUTO NAME
 SELECT snapshotId, snapshotAt, waiter, who, what, object_id,
record_id,
        owner, is_victim, rollback_operation_count
    FROM sa_report_deadlocks ();
 COMMIT;
 CALL xp_startmail ( mail_user ='John Smith',
                     mail_password ='mypwd' );
 CALL xp_sendmail( recipient='DBAdmin',
                   subject='Deadlock details added to the
DeadlockDetails table.' );
```

```
 CALL xp_stopmail ( );
END;
```

This event copies the results of the **sa_report_deadlocks** system procedure into a table and notifies the administrator about the deadlock.

3. Set the **log_deadlocks** option on.
```
SET OPTION PUBLIC.log_deadlocks = 'On';
```
4. Enable logging of the most-recently executed statement.
```
CALL sa_server_option( 'RememberLastStatement', 'YES' );
```

sa_report_deadlocks System Procedure

Retrieves information about deadlocks from an internal buffer created by the database server.

Syntax

sa_report_deadlocks( )

Result set

Column Name	Data Type	Description
snapshotId	BIGINT	The deadlock instance (all rows pertaining to a particular deadlock have the same ID).
snapshotAt	TIMESTAMP	The time when the deadlock occurred.
waiter	INT	The connection handle of the waiting connection.
who	VARCHAR(128)	The user ID associated with the connection that is waiting.
what	LONG VARCHAR	The command being executed by the waiting connection. This information is only available if you have turned on capturing of the most recently-prepared SQL statement by specifying the **-zl** option on the database server command line.
object_id	UNSIGNED BIGINT	The object ID of the table containing the row.
record_id	BIGINT	The row ID for system tables

Column Name	Data Type	Description
owner	INT	The connection handle of the connection owning the lock being waited on.
is_victim	BIT	Identifies the rolled back transaction.
rollback_operation_count	UNSIGNED INT	The number of uncommitted operations that may be lost if the transaction rolls back.
iq_rid	UNSIGNED BIGINT	The row ID for IQ RLV enabled tables.
iq_txn_id	UNSIGNED BIGINT	The transaction id of the associated row

Remarks
When the log_deadlocks option is set to On, the database server logs information about deadlocks in an internal buffer. You can view the information in the log using the sa_report_deadlocks system procedure.

Privileges
You must have the MONITOR system privilege.

Side effects
None.

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

sa_conn_info system procedure

Reports connection property information.

Syntax

```
sa_conn_info( [ connidparm ] )
```

Arguments

- ***connidparm* –** This optional INTEGER parameter specifies the connection ID number. The default is NULL.

Result set

Column name	Data type	Description
Number	INTEGER	Returns the connection ID (a number) for the current connection.
Name	VARCHAR(255)	Returns the connection ID (a number) for the current connection. Temporary connection names have `INT:` prepended to the connection name.
Userid	VARCHAR(255)	Returns the user ID for the connection.
DBNumber	INTEGER	Returns the ID number of the database.

Column name	Data type	Description
LastReqTime	VARCHAR(255)	Returns the time at which the last request for the specified connection started. This property can return an empty string for internal connections, such as events.
ReqType	VARCHAR(255)	Returns the type of the last request. If a connection has been cached by connection pooling, its ReqType value is CONNECT_POOL_CACHE.
CommLink	VARCHAR(255)	Returns the communication link for the connection. This is one of the network protocols supported by SAP Sybase IQ, or local for a same-computer connection.
NodeAddr	VARCHAR(255)	Returns the address of the client in a client/server connection.
ClientPort	INTEGER	Returns the client's TCP/IP port number or 0 if the connection isn't a TCP/IP connection.
ServerPort	INTEGER	Returns the database server's TCP/IP port number or 0.
BlockedOn	INTEGER	Returns zero if the current connection isn't blocked, or if it is blocked, the connection number on which the connection is blocked because of a locking conflict.

Column name	Data type	Description
LockRowID	UNSIGNED BIGINT	Returns the identifier of the locked row. LockRowID is NULL if the connection is not waiting on a lock associated with a row (that is, it is not waiting on a lock, or it is waiting on a lock that has no associated row).
LockIndexID	INTEGER	Returns the identifier of the locked index. LockIndexID is -1 if the lock is associated with all indexes on the table in LockTable. LockIndexID is NULL if the connection is not waiting on a lock associated with an index (that is, it is not waiting on a lock, or it is waiting on a lock that has no associated index).
LockTable	VARCHAR(255)	Returns the name of the table associated with a lock if the connection is currently waiting for a lock. Otherwise, LockTable returns an empty string.
UncommitOps	INTEGER	Returns the number of uncommitted operations.
ParentConnection	INTEGER	Returns the connection ID of the connection that created a temporary connection to perform a database operation (such as performing a backup or creating a database). For other types of connections, this property returns NULL.

Remarks

If *connidparm* is less than zero, then a result set consisting of connection properties for the current connection is returned. If *connidparm* is not supplied or is NULL, then connection properties are returned for all connections to all databases running on the database server.

In a block situation, the BlockedOn value returned by this procedure allows you to check which users are blocked, and who they are blocked on. The sa_locks system procedure can be used to display the locks held by the blocking connection.

For more information based on any of these properties, you can execute something similar to the following:

```
SELECT *, DB_NAME( DBNumber ),
   CONNECTION_PROPERTY( 'LastStatement', Number )
   FROM sa_conn_info( );
```

The value of LockRowID can be used to look up a lock in the output of the sa_locks procedure.

The value in LockIndexID can be used to look up a lock in the output of the sa_locks procedure. Also, the value in LockIndexID corresponds to the primary key of the ISYSIDX system table, which can be viewed using the SYSIDX system view.

Every lock has an associated table, so the value of LockTable can be used to unambiguously determine whether a connection is waiting on a lock.

Privileges

No privileges are required to execute this system procedure for the current connection ID. To execute this system procedure for other connections, you must have either the SERVER OPERATOR, MONITOR, or DROP CONNECTION system privilege.

Side effects

None

Examples

The following example uses the sa_conn_info system procedure to return a result set summarizing connection properties for all connections to the server.

```
CALL sa_conn_info( );
```

Number	Name	Userid	DBNumber	...
79	SQL_DBC_10dcf810	DBA	0	...
46	setup	User1	0	...
...	...	...	...	...

The following example uses the sa_conn_info system procedure to return a result set showing which connection created a temporary connection.

```
SELECT Number, Name, ParentConnection FROM sa_conn_info();
```

Connection 8 created the temporary connection that executed a CREATE DATABASE statement.

```
Number      Name                 ParentConnection
------------------------------------------------
```

```
1000000048  INT: CreateDB       8
9           SQL_DBC_14675af8    (NULL)
8           SQL_DBA_152d5ac0    (NULL)
```

Versions

The database server binds the snapshot version when a transaction is created, after the transaction has obtained the necessary write locks on the table or the table row. Table-level versioning enables single-writer access and table-level locking. Row-level versioning (RLV) enables concurrent writer access and row-level locking for RLV-enabled tables.

Table-Level Snapshot Versioning

The default transaction processing behavior, *table-level snapshot versioning*, allows multiple writers to modify a table serially—that is, one after the other, never more than one at a time—while multiple readers continue to work on an original copy of the table.

Without table-level snapshot versioning, concurrent read and write operations might cause inconsistencies in the IQ main store. The table-level versioning provided by SAP Sybase IQ prevents inconsistencies both by serializing transactions, and by making the table the version level.

While any transaction processing system is designed to ensure that the database remains consistent, the SAP Sybase IQ approach means that users need not worry about placing their queries and updates in appropriate transactions. SAP Sybase IQ begins and ends transactions automatically, and ensures that read and write operations do not interfere with each other.

When table-level snapshot versioning is enabled but the table is configured for RLV storage, concurrent read and write operations are still disallowed. The table-level snapshot versioning setting overrides a table's RLV setting.

Enabling Table-Level Snapshot Versioning

Table-level snapshot versioning is enabled by default. If the default versioning method has been changed, use the **ALLOW_SNAPSHOT_VERSIONING** database option to enable it. Temporarily set table-level versioning by issuing:

```
SET TEMPORARY OPTION Snapshot_Versioning = 'Table-level'
```

Table-Level Locking

All table-level locks occur automatically, based on the type of operation a user requests.

You need not explicity request a lock; the transaction that has access to the table holds the lock.

When a table is locked in SAP Sybase IQ, no other transaction, except data definition operations, can have write access to it. Any other write transaction that attempts to access a table with a write lock on it receives an error. Any transaction can obtain read access.

The locks maintain the reliability of information in the database by preventing concurrent access by other transactions. The database server retains all the locks acquired by a transaction until the transaction completes, due to either a commit or a rollback.

You can reserve write locks on a set of tables within a new transaction using the **LOCK TABLE** statement. **LOCK TABLE** commits the current transaction and allows transactions to enqueue until the locks are available.

Table Locks for DML Operations

Data manipulation language (DML) operations include insertions, deletions, updates, and queries. For all such operations, table-level snapshot versioning permits one writer and multiple readers on any given table.

In table-level snapshot versioning:

- Read transactions do not block write transactions.
- Write transactions do not block read transactions.
- A single update user and multiple read-only users can concurrently access a table.
- Only one user can update the data in a given table at one time.

The first transaction to open a table in write mode gains access to the table. A second transaction that tries to open the table in write mode receives an error. Any additional attempts to write to the table in the current transaction fail. The transaction can continue, but only with read operations or with writes to other tables.

SAP Sybase IQ supports share, write, and exclusive lock enqueuing, allowing you to lock a table for a specified period. You can write lock multiple tables simultaneously.

To avoid future version errors from subsequent DML statements, use the **LOCK TABLE** statement to reserve a write lock on the table or set of tables that you plan to modify.

In the case of deadlocks, the last **LOCK TABLE** statement that became blocked is usually rolled back and an error returns to that transaction about the form of deadlock that occurred.

In certain cases, you must issue **COMMIT** or **ROLLBACK** statements. For example, an explicit **COMMIT** or **ROLLBACK** is required to release locks when DML statements fail due to integrity constraints.

If a DML statement fails due to a referential integrity violation on the referenced table or an unavailable lock on other tables, you see: SQL Anywhere Error -210.

Table Locks for DDL Operations

Data definition language (DDL) operations include **CREATE**, **DROP**, and **ALTER**.

DDL operations on a given table or index lock out all other readers and writers from any table being modified. This approach is crucial to the accuracy of query results. It ensures, for example, that a table column does not disappear from the database while you are selecting data from that column.

CREATE, **DROP**, and **ALTER** commands have special properties:

- They cannot start while any other transaction is using the table or index they are modifying.

  For example, if a user issues a SELECT on a table, the table is locked and cannot be altered until the user logs out, issues a SELECT on another table, or issues a ROLLBACK.
- They include an automatic **COMMIT** on completion.
- Existing transactions that try to use the table being modified receive an error. In other words, if you are accessing a table, and a DDL command changes that table, your command fails.
- At any given time, only one of the commands **CREATE DBSPACE**, **DROP DBSPACE**, and **CHECKPOINT** can be executing in a database.

DDL Locking Errors

Errors may occur if you issue a DDL command when another DDL command is in process.

You may see this error message:

```
Cannot perform DDL command now on table <tablename> as a DDL command
is already in progress on that table.
```

If a **CREATE DBSPACE** or **DROP DBSPACE** command is in progress, and a user explicitly issues a **CHECKPOINT** command, the checkpoint fails with:

```
Run time SQL Error
```

If a **CHECKPOINT** command is in progress, a user who issues a **CREATE DBSPACE** or **DROP DBSPACE** command sees:

```
Cannot perform requested command as there is a
CHECKPOINT command in progress.
```

A user who issues **CREATE DBSPACE** during a drop sees:

```
Cannot perform requested command as there is a
DROP DBSPACE command in progress.
```

A user who issues **DROP DBSPACE** during a create sees:

```
Cannot perform requested command as there is a
CREATE DBSPACE command in progress.
```

When one transaction issues a DDL command on a given table or index, any other transaction that began before the DDL transaction commits, and that tries to access that table, receives an error. When this error occurs, any additional attempts to read or write to the table in the current transaction fail.

There is an exception to these index creation commands rules. If you use a **CREATE INDEX** command with a **SELECT** statement on the table(s) affected by the index creation, SAP Sybase IQ prevents use of the new index until the transaction creating the index commits.

While **GRANT**, **REVOKE**, and **SET OPTION** are also considered DDL operations, they cause no concurrency conflicts, and so are not restricted. **GRANT** and **REVOKE** always cause an automatic commit; **SET OPTION** causes an automatic commit except when it is specified as **TEMPORARY**. **GRANT** and **REVOKE** are not allowed for any user currently connected to the

database. **SET OPTION** affects all subsequent SQL statements sent to the database server, except for certain options that do not take effect until after you restart the database server.

Primary Keys and Locking

Because only one user can update a table when table-level snapshot versioning is enabled, primary key generation does not cause concurrency conflicts.

Displaying Active Table Locks

Identify the user who has a table locked. An attempt to write to a table fails if another transaction holds a lock on that table.

1. Run **sp_iqtransaction**.
2. In the output, find the transaction identifier and the user name.
3. Run the **sp_iqlocks** procedure.

   For each lock in the catalog store and the IQ store of your current database, **sp_iqlocks** tells you:
   - The connection and user ID that holds the lock
   - The table on which the lock is held
   - The type of lock, and a name to identify the lock

For example, when another transaction holds a lock, you see:

```
Cannot open the requested object for write in the current transaction
(TxnID1). Another user has write access in transaction TxnID2.
```

Find TxnID2 in the output of **sp_iqtransaction**, and look for the name of the user in the same row of output.

Performance Implications of Table-Level Snapshot Versioning

Typically, table-level snapshot versioning has a minimal impact on performance.

However, there are certain resource issues you should be aware of:

- Buffer consumption may increase slightly if multiple users are using different versions of the same database page simultaneously.
- Version management requires some overhead, but the effect on performance is minimal.
- The thread control, which determines how many processing resources a user gets, and the sweeper controls, which use a small number of threads to sweep dirty data pages out to disk, have a minor impact on performance.
- Disk space can sometimes become an issue. Storing overlapping versions has the potential to use a lot of disk space, depending on the number and size of versions in use simultaneously. Metadata and database page versions are retained until they are dropped, either at a **RELEASE SAVEPOINT** or when the last transaction that can see a given version commits or rolls back. The space is then reclaimed.

Delays due to locking are minimal. Individual commits, rollbacks, and checkpoints might block other read or write transactions, but only very briefly.

All of these performance and disk use factors only affect your system in the degree to which you take advantage of the SAP Sybase IQ concurrent read and write capabilities. Disk space requirements in particular can vary widely, depending on how long write transactions take before they commit, how many read transactions take place during write transactions, the number of rows these transactions affect, and whether you allow the release of data pages at interim savepoints.

Overlapping Versions and Deletions

To delete data when using table-level versioning, you may actually need to increase disk space by adding a dbspace to your IQ main store.

The amount of space you need for a deletion depends on the distribution of the data on data pages, more than on the size or number of rows being deleted. SAP Sybase IQ must retain a version of each page that contains any of the data you are deleting, from the time the deletion begins until the transaction commits. If the rows being deleted are distributed across many data pages, you need space in your IQ main store to retain all of those extra data pages.

For example, say you must delete 10 rows from a database where each page holds 100 rows. If each of those 10 rows is on a separate data page, then your IQ main store must have space for 10 version pages, each big enough to hold 100 rows. This distribution is unlikely, but it is possible.

The space needed to delete data varies by index type. It is proportional to—and in the worst case, equal to—the size of the index from which you are deleting.

If you run out of space while deleting data, SAP Sybase IQ halts the deletion and writes this message in the notification log:

```
Out of disk space
```

After you add space, the deletion resumes. When the delete transaction commits, the space becomes available for other deletions or insertions. If you do not normally need that much space in your database, you can drop the dbspace to regain the extra disk space for other purposes. Do so before inserting any data that may need the new dbspace.

Running out of space during a deletion should not affect other query users.

If you run out of space, but do not have enough disk space to add another dbspace, you must shut down the database engine and then restart it, allowing the database to roll back. You can then delete the rows in smaller, separate transactions.

Note: DROP TABLE and **DROP DATABASE** delete the table or database and all data in it without creating any version pages, so you do not need to add space to use these commands.

Row-Level Snapshot Versioning

Row-level snapshot versioning applies only to tables enabled for in-memory RLV storage. Row-level snapshot versioning allows multiple writers to make concurrent DML changes to a table, but never to the same rows at the same time.

Row-level snapshot versioning locks the table at the row level using row locks. A row lock provides a write lock for a table row, meaning the transaction gets blocked, or fails, depending on the **BLOCKING** and **BLOCKING_TIMEOUT** option settings. If **BLOCKING** is ON, the transaction blocks. If **BLOCKING** is OFF, the transaction fails immediately with an `ALREADY LOCKED` SQL exception.

Transaction blocking enables row-level snapshot versioning to write to different rows of the same table simultaneously. Depending on the **BLOCKING** and **BLOCKING_TIMEOUT** option settings, row-lock contention results either in an error, or a retry to obtain the lock if it is released within the specified timeout period. When a transaction configured for table-level versioning attempts to write to a table with a row locked by a row-level versioned transaction, the table-level transaction either fails with an error, or blocks and retries if the lock is released within the specified timeout period.

DDL changes to a table (**CREATE**, **DROP**, and **ALTER**), however, lock the table at the table level.

Specifying Snapshot Versioning

Use the **SNAPSHOT_VERSIONING** option to set the snapshot versioning type to either `Row-level` or `Table-Level`. You can set the option at the database (PUBLIC) level, connection level (TEMPORARY) or user level. To use the in-memory RLV store, enable row-level snapshot versioning for your transactions. For simultaneous updates to different rows of the same table, each transaction or connection must also enable row-level snapshot versioning.

Prerequisites

- If setting to Row-level, the RLV store dbspace exists with at least one dbfile.
- If setting to Row-level, the table is RLV-enabled.
- Requires the SET ANY PUBLIC OPTION system privilege to set this option for PUBLIC or for other user or role.

Task

Once the snapshot versioning property has been set for a transaction, it remains the same until the transaction commits.

1. Determine the scope of the **SET OPTION** command to set the option as a database-wide option, connection-level option, or user-level option:

- **SET OPTION public.SNAPSHOT_VERSIONING...**
- **SET TEMPORARY OPTION SNAPSHOT_VERSIONING...**
- **SET OPTION** *username*.**SNAPSHOT_VERSIONING...**

2. Specify the snapshot versioning type.

Level	Option
Row-level	Row-level snapshot versioning. Required for in-memory RLV storage. Row-level snapshot versioning allows multiple writers to make concurrent DML changes to a table, but never to the same rows at the same time.
Table-level	Classic (backward-compatible) SAP Sybase IQ versioning behavior. Takes snapshots at the table-level. Multiple writers cannot make concurrent DML changes to a table.

```
SET TEMPORARY OPTION Snapshot_Versioning = 'Row-level';

CREATE TABLE rv_locks(c1 int, c2 int, c3 int);

INSERT INTO rv_locks VALUES (1,1,1);
INSERT INTO rv_locks VALUES (2,2,2);
INSERT INTO rv_locks VALUES (3,3,3);
INSERT INTO rv_locks VALUES (4,4,4);
COMMIT;
```

Tutorial: Monitoring Write-Intent Locks

In this tutorial, create RLV-enabled tables, execute a transaction, and use the **sp_iqlocks** stored procedure to report on schema-level locks and write-intent locks in the database. Then use the **sp_iqconnection** and **sa_conn_info** stored procedures to view the internal connection controlling the write-intent lock.

Prerequisites

- SAP Sybase IQ server has a simplex database.
- RLV storage is configured.

Task

Tip: You can monitor locks using Sybase Control Center. For more information, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

1. Create RLV-enabled tables `rv_locks` and `rv_locks2`, and configure table-level snapshot versioning.

```
SET TEMPORARY OPTION SNAPSHOT_VERSIONING = 'Table-level';

CREATE TABLE rv_locks(c1 INT, c2 INT, c3 INT);
```

```
INSERT INTO rv_locks VALUES (1,1,1);
INSERT INTO rv_locks VALUES (2,2,2);
INSERT INTO rv_locks VALUES (3,3,3);
INSERT INTO rv_locks VALUES (4,4,4);
COMMIT;

CREATE TABLE rv_locks2(c1 int, c2 int, c3 int);

INSERT INTO rv_locks2 VALUES (1,1,1);
INSERT INTO rv_locks2 VALUES (2,2,2);
INSERT INTO rv_locks2 VALUES (3,3,3);
INSERT INTO rv_locks2 VALUES (4,4,4);
COMMIT;

ALTER TABLE rv_locks ENABLE RLV STORE;
ALTER TABLE rv_locks2 ENABLE RLV STORE;
```

2. Enable connection blocking and set the blocking timeout threshold:

```
SET TEMPORARY OPTION BLOCKING = 'ON';
SET TEMPORARY OPTION BLOCKING_TIMEOUT = '0';
```

3. Use the **sp_iqlocks** stored procedure to view the current set of database locks. At this point, no locks are returned.

```
sp_iqlocks
```

   The absence of a write-intent lock for the RLV-enabled table indicates that the in-memory RLV portion of the table has yet to be created.

4. Set the snapshot versioning property of the transaction to row-level.

```
SET TEMPORARY OPTION SNAPSHOT_VERSIONING = 'row-level';
```

5. Write to the table.

```
INSERT INTO rv_locks VALUES (5,5,5);
```

   Writing to, or querying, an RLV-enabled table creates the RLV-enabled portion of the table in memory, on demand.

6. Re-execute **sp_iqlocks**.

```
sp_iqlocks
```

   This time, the procedure returns a write-intent lock.

```
conn_name,conn_id,user_id,table_type,creator,table_name,index_id,
lock_class,lock_duration,lock_type,row_identifier,row_range
'SQL_DBC_13cd6038',
3,'DBA','BASE','DBA','rv_locks',,'Schema','Transaction','Shared',
,
'RLV_CONN_T775',
1000000407,'','BASE','DBA','rv_locks',,'Table','Transaction','Int
ent',,
```

   Connection ID 100000407 has a write-intent lock on the `rv_locks` table. The lock type is set to `Intent`, which indicates a write intent lock.

Note: The connection ID number (100000407) is large because it represents an internal connection within the server itself. This internal connection is used to manage locks on the RLV-enabled table.

ConnectionID 3 has a schema lock on the table. The lock type is set to `Shared`, which indicates a shared schema lock

7. Return to the uncommitted transaction that performed the insert, and commit it:

```
Commit
```

During the commit, the database releases the locks held by the transaction. For the tutorial, this releases only the shared schema lock. The RLV-enabled table now exists in memory, with committed data. Therefore, the only lock present at this point is the write-intent lock held by the RLV-enabled portion of the table.

8. Re-execute **sp_iqlocks**.

```
sp_iqlocks
```

The schema lock is gone, but the write-intent lock remains:

```
conn_name,conn_id,user_id,table_type,creator,table_name,index_id,
lock_class,lock_duration,lock_type,row_identifier,row_range
'RVL_CONN_T775',
1000000407,'','BASE','DBA','rv_locks2',,'Table','Transaction','In
tent',,
```

Note: The row for conn_id 100000407 has not changed since the last time you executed **sp_iqlocks**.

9. Execute **sp_iqconnection** to view connection details

```
sp_iqconnection
```

You see:

```
ConnHandle,Name,Userid,LastReqTime,ReqType,IQCmdType,LastIQCmdTim
e,IQCursors,LowestIQCursorState,IQthreads,TxnID,ConnCreateTime,Te
mpTableSpaceKB,TempWorkSpaceKB,IQconnID,satoiq_count,iqtosa_count
,CommLink,NodeAddr,LastIdle,MPXServerName,LSName,INCConnName,INCC
onnSuspended
1,'SQL_DBC_13de5fd8','DBA','2012-08-08
08:49:25.629','PREFETCH','NONE',2012-08-08 08:49:25.0,0,'NONE',
0,0,2012-08-08 08:49:24.0,0,0,70,40,2,'local','',0,,,'','N'
3,'SQL_DBC_13cd6038','DBA','2012-08-08
09:25:32.920','OPEN','IQUTILITYOPENCURSOR',2012-08-08
09:25:32.0,0,'NONE',0,1008,2012-08-08
08:50:04.0,0,0,92,187,413,'local','',8789,,,'','N'
1000000407,'INT: RLVLockConn','','','unknown (0)','NONE',
0001-01-01 00:00:00.0,0,'NONE',0,0,2012-08-08
09:00:40.0,0,0,410,2,0,'NA','NA',0,,,'','N'
```

The third row (ConnHandle 1000000407) provides information on the internal connection (RLVLockConn) used by the RLV-enabled table to control the write-intent lock.

Note: ConnHandle 1000000407 matches conn_id 100000407 in **sp_iqlocks** output. It also matches ConnHandle 1000000407 in **sp_iqtransaction** output.

10. Execute **sa_conn_info** to view additional connection details. **sa_conn_info** is similar to **sp_iqconnection**.

```
sa_conn_info
```

You see:

```
Number,Name,Userid,DBNumber,LastReqTime,ReqType,CommLink,NodeAddr
,ClientPort,ServerPort,BlockedOn,LockRowID,LockIndexID,LockTable,
UncommitOps,ParentConnection
1000000407,sa_'INT: RLVLockConn','',0,'','unknown (0)','NA','NA',
0,0,0,0,,'',0,
3,'SQL_DBC_13cd6038','DBA',0,'2012-08-08
09:30:43.799','FETCH','local','',0,0,0,0,,'',0,
1,'SQL_DBC_13de5fd8','DBA',0,'2012-08-08
08:49:25.629','PREFETCH','local','',0,0,0,0,,'',0,
```

Note: In the first row, Number 1000000407 matches ConnHandle 1000000407 in the **sp_iqconnection** output, and conn_id 100000407 in the **sp_iqlocks** output.

Userid "INT: RLVLockConn" indicates an internal connection. This connection is used by the RLV-enabled table to control the write-intent lock.

Row Locks

A row lock is a table-row write lock that allows the holding transaction to write to any column of a locked row. Only one holder of this lock can exist at a time. A *write-intent* lock is a prerequisite; the transaction must hold a write-intent lock before the lock manager grants it a row lock.

A table-row write lock allows the holding transaction to write to any column of locked row. This lock cannot be granted without the requesting transaction first holding the write intent lock. Row write locks are exclusive locks; only one transaction can hold a write lock on a row at any time. Once a transaction acquires a write lock, requests to lock the row by other transactions are denied.

Row locks exist only during row deletions. The RLV store is an append-only store, meaning that every write action results in a new row appended to the store. **INSERT** statements append a new row to the store, as do **UPDATE** statements. The RLV store considers an **UPDATE** to be a **DELETE** followed by an **INSERT**. Before a row is deleted, either in the context of a **DELETE** or **UPDATE** statement, the database takes out a row-level lock.

Monitoring Write Intent Locks in Interactive SQL

View details on write intent locks using the **sa_locks** system procedure.

Prerequisites

- RLV storage is configured.

- A table is registered for RLV storage.
- The database option SNAPSHOT_VERSIONING is set to 'Row-level.'

Task

1. Run the **sa_locks** system procedure.
2. View the lock_class column, which shows intent locks for tables and rows.
3. Run the **sp_iqlocks** stored procedure to identify users who are blocking other users from writing to a table. This procedure displays information about locks currently held in the database, including the connection and user ID that holds the lock, the table on which the lock is held, the type of lock, and a name to identify the lock.

Write-Intent Locks

A write-intent lock is a table write lock that grants the transaction permission to write to a table row in the future. A write-intent lock can be held by multiple requesting connections.

A write intent lock always exists when the RLV-enabled portion of the table exists in memory. You can view details of the write intent lock using the **sp_iqlocks** stored procedure.

The write-intent lock conflicts with table write locks and table exclusive locks. This conflict prevents a table-level snapshot-versioned transaction from writing to the table or performing a DDL operation until the lock manager releases all write-intent locks on the table. In a situation where both table-level snapshot-versioned transactions and row-level snapshot-versioned transaction connections write to a table, write-intent locks provide synchronization. Consider this scenario:

Connection	Action
Row-level snapshot-versioned transaction A	• Executes query writing to multiple rows of table_1. • Lock manager creates a write-intent lock for table_1. • Lock manager creates multiple local write-intent locks for row-level DML updates. Lock manager creates row-level locks.
Table-level snapshot-versioned transaction B	Attempts to write to table_1. Transaction B blocked by write intent lock.
Row-level snapshot-versioned transaction A	Commits transaction A. Table changes are merged from the RLV store to the IQ main store. Write-intent locks released.
Table-level snapshot-versioned transaction B	Proceeds with write to table_1.

Row-Level DDL Locking Considerations

Data Definition Language (DDL) changes (for example, **CREATE INDEX**, **DROP INDEX**, and **ALTER TABLE** ADD, ALTER, or DROP) to an RLV-enabled table require an exclusive table-

level lock. For DDL events, the locking behavior for an RLV-enabled table is the same as for an IQ main store table: the writing connection has an exclusive lock on the table. When **BLOCKING** is set to ON, all competing DML and DDL transactions against the table are blocked until the DDL changes are committed. When **BLOCKING** is set to OFF, the competing transaction will immediately fail the lock request.

Viewing the Versioning Type of a Transaction

Use the **sp_iqtransaction** stored procedure to view the versioning type (either table-level or row-level) of each active transaction.

1. Run **sp_iqtransaction**.
2. Examine the VersioningType column.

VersioningType	Description
Row-level	The transaction uses row-level versioning. Row-level versioning enables concurrent writer access and row-level locking for RLV-enabled tables.
Table-level	The transaction uses table-level versioning. Table-level versioning enables single-writer access and table-level locking.

See also

- *sp_iqtransaction Procedure* on page 218

Temporary Table Versioning

A temporary table that is created in the database is called a *global temporary table*. A temporary table that is declared (rather than created in the database) is called a *local temporary table*.

A global temporary table is accessible to all users with the appropriate permissions. Each user has his or her own instance of the table, however; only one user ever sees a given set of rows. By default, the rows of a global temporary table are deleted on **COMMIT**. You can override this default, by specifying **ON COMMIT PRESERVE ROWS** when you create the temporary table.

Only one user sees any of the rows in a local temporary table. The table is dropped when that user disconnects. When you declare a local temporary table, SAP Sybase IQ issues a savepoint instead of committing the transaction automatically, as it would for a data definition operation on any other type of table. Before creating an index, commit the data in the local temporary table. If you attempt to create an index using uncommitted data, you may see: "`Local temporary table, <tablename>, must be committed in order to create an index.`"

SAP Sybase IQ makes no distinction between versioning base tables (IQ main store database tables) and versioning global temporary tables. Because the data in any temporary table is

accessible to only one user, there will never be more than one write transaction open for a temporary table.

You can enable a global temporary table or local temporary table for RLV storage, even though a temporary table cannot take advantage of RLV's multiple user concurrent write capability. Enabling a temporary table for RLV storage lets you take advantage of the RLV store's low-latency DML.

Investigating Lock Contention Effects on Performance

Some load or query performance issues result from lock contention. If your kernel system time is greater than 10%, you may be experiencing lock contention.

1. To find out if lock contention is affecting performance on your system, do one of the following:
   - Run the SAP Sybase IQ monitor with the **-contention** option.
   - On UNIX platforms, run the **sar** or **vmstat** utility.
   - On Windows platforms, check the CPU usage in the Task Manager.
2. If you suspect lock contention, you may find it useful to control the level of partitioning directly by setting:
   - The **-iqpartition** server startup option.
   - The **cache_partitions** database option.

   Note: Higher than normal kernel system time can also indicate that your kernel is not well tuned. If this is the case, you may need to adjust kernel parameters; changing SAP Sybase IQ settings does not fix an improperly tuned kernel.

See also

- *CACHE_PARTITIONS Option* on page 13

(deprecated)-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

```
monitor_options -contention
```

Output

Table 21. -contention Output Fields

Output Field	Description
AU	Current number of active users
LRULks	Number times the LRU was locked (repeated for the temp cache)
woTO	Number times lock was granted without timeout (repeated for the temp cache)
Loops	Number times SAP Sybase IQ retried before lock was granted (repeated for the temp cache)
TOs	Number of times SAP Sybase IQ timed out and had to wait for the lock (repeated for the temp cache)
BWaits	Number of busy waits for a buffer in the cache (repeated for the temp cache)
IOLock	Number of times SAP Sybase IQ locked the compressed I/O pool (repeated for the temp cache); can be ignored
IOWait	Number of times SAP Sybase IQ had to wait for the lock on the compressed I/O pool (repeated for the temp cache); can be ignored
HTLock	Number of times SAP Sybase IQ locked the block maps hash table (repeated for the temp cache)
HTWait	Number of times SAP Sybase IQ had to wait for the block maps hash table (repeated for the temp cache); HTLock and HTWait indicate how many block maps you are using
FLLock	Number of times SAP Sybase IQ had to lock the free list (repeated for the temp cache)
FLWait	Number of times SAP Sybase IQ had to wait for the lock on the free list (repeated for the temp cache)
MemLks	Number of timesSAP Sybase IQ took the memory manager (heap) lock
MemWts	Number of times SAP Sybase IQ had to wait for the memory manager lock

Note: SAP Sybase IQ no longer uses spin locks. As a result, woTO, Loops, and TOs statistics are rarely used.

Checkpoints, Savepoints, and Transaction Rollback

Besides permitting concurrency, snapshot versioning transaction processing plays an important role in data recovery.

SAP Sybase IQ relies on transaction-related commands that help you recover a stable set of data in the event of system or media failure. The **CHECKPOINT** command sets a checkpoint, the **SAVEPOINT** command sets a savepoint, **RELEASE SAVEPOINT** releases savepoints, and **ROLLBACK TRANSACTION** rolls back transactions.

Checkpoints

A checkpoint marks a significant point in a transaction, when SAP Sybase IQ writes to disk certain information it tracks internally. SAP Sybase IQ uses this information during database recovery.

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.

Most checkpoints occur automatically. You can also set explicit checkpoints, although you do not need to do so.

A checkpoint occurs:

- When a transaction issues a **CHECKPOINT** command
- When the **CHECKPOINT_TIME** is exceeded
- At the start and end of the backup process
- When the database server is shut down

The **CHECKPOINT_TIME** is the maximum time that can pass between checkpoints. By default, it is 60 minutes. To adjust the checkpoint interval, use the **SET OPTION** statement. 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.

Savepoints Within Transactions

SAP Sybase IQ supports savepoints within a transaction.

A **SAVEPOINT** statement defines an intermediate point during a transaction. Because a single IQ transaction may write millions of rows of data, you may want to limit the amount of data that is committed—and thus written to the IQ main store—to less than a full transaction's worth. Setting savepoints allows you to subdivide transactions.

Releasing a Savepoint

Releasing a savepoint frees up the version pages that have been used up to that savepoint.

Data is versioned internally at the page level. SAP Sybase IQ maintains a separate copy of only the updated pages; the remaining pages are shared with the previous version.

Releasing savepoints makes better use of your disk space.

This Interactive SQL command releases savepoint *n*:

```
RELEASE SAVEPOINT n
```

All resources after the savepoint are released, and you cannot roll back to any intermediate savepoints.

RELEASE SAVEPOINT does not release locks.

Rollbacks to Savepoints

A **ROLLBACK TO SAVEPOINT** command undoes all changes after a savepoint.

This command rolls back to the savepoint you specify, or to the most recent savepoint if you do not specify a named savepoint. Rolling back to savepoint *n* undoes all actions for all savepoints greater than or equal to *n*.

Normally, locks are released only at the end of a transaction. However, **ROLLBACK TO SAVEPOINT** does release locks under certain conditions, as in the following scenario:

Assume you have a series of savepoints in a transaction, and you then perform a write operation. Roll back the transaction to an earlier savepoint. The rollback undoes all actions after that savepoint, including the write operation and any locks it acquires after the savepoint you are rolling back to.

SAP Sybase IQ supports savepoint operations on updatable cursors.

Automatic and User-Defined Savepoints

IQ sets an implicit savepoint before and after every DML command.

The data page versions associated with these savepoints are released when the command completes. To retain data page versions beyond the end of a single DML command, set your own, named savepoints.

Named and Nested Savepoints

Named, nested savepoints provide many active savepoints within a transaction.

You can cancel changes between a **SAVEPOINT** and a **RELEASE SAVEPOINT** by rolling back to a previous savepoint or rolling back the transaction itself. Changes within a transaction are not a permanent part of the database until the transaction is committed. All savepoints are released when a transaction ends.

Savepoints cause SAP Sybase IQ to update information it maintains about the location of available disk space. This information is used during transaction rollback.

There is no additional overhead in using savepoints, although unreleased savepoints may consume extra disk space by keeping older intermediate versions active.

Transaction Rollback

When you roll back a transaction, either automatically, or by explicit user request, you undo all of the operations in that transaction.

Rolling back the database means returning the database to an earlier state.

Cause of Rollback

Rollbacks can occur either due to an explicit user request, or automatically.

Use a **ROLLBACK** statement to undo any changes to the database since the last **COMMIT** or **ROLLBACK**.

Use a **ROLLBACK TO SAVEPOINT**statement to undo any changes to the database since the **SAVEPOINT** you name, or else to the last **SAVEPOINT**.

SAP Sybase IQ rolls back the database automatically if a user is in a transaction and then logs out or disconnects without committing. The rollback is to the most recent commit or rollback.

Effect of Rollback

Rollback returns both the main and temporary stores to their former state, and also releases locks.

- Transaction rollback releases all locks held by the transaction.
- Rollback to a savepoint releases all locks acquired after that savepoint.

Rolling back open cursors deletes all cursor information and closes both hold and non-hold cursors:

- Transaction rollback closes all cursors, regardless of whether the cursor was opened in the transaction being rolled back, or in an earlier transaction.
- Rollback to a savepoint closes all cursors opened after that savepoint.

System Recovery

In the event of a system failure or power outage, or when you restart the database server after it has been stopped, SAP Sybase IQ attempts to recover automatically.

During SAP Sybase IQ database recovery, any uncommitted transactions are rolled back, and any disk space used for old versions is returned to the pool of available space. At this point, the database contains only the most recently committed version of each permanent table.

During recovery from a system failure, SAP Sybase IQ reopens all connections that were active at the time of the failure. If the **-gm** parameter, which sets the number of user connections, was in effect at the time of the failure, restart the IQ server with at least as many connections as were in use when the failure occurred. Temporary table contents cannot be recovered.

If a failure occurs, try to restart the database server and database. You will need information from your server log and IQ message log to recover.

Run the stored procedure **sp_iqcheckdb** after a system failure, preferably before allowing users to connect. This procedure checks every block in your database, and produces statistics that allow you to check the consistency and integrity of your database.

How Transaction Information Aids Recovery

The SAP Sybase IQ recovery mechanism is designed for a data warehouse enivronment, in which, typically, transactions are few but lengthy.

SAP Sybase IQ performs database updates on a copy of the actual database page, and then writes the data to disk whenever a write transaction commits. SAP Sybase IQ also records:

- The location and quantity of changed data for each transaction is stored in a *transaction log*.
- The location of any version pages and free space is stored on disk. SAP Sybase IQ uses this information to free up space when versions are no longer needed. All versions created during a write transaction become obsolete when the write transaction commits or rolls back. Individual versions can be released at a savepoint.
- Additional information about checkpoints that occurred during a transaction.

When you need to recover your database, SAP Sybase IQ restores quickly from the information in the transaction log and the checkpoint information. It uses the information about versions and free space to roll back transactions, and to release the disk space occupied by obsolete versions.

The transaction log requires very little space: only about 128 bytes for each committed transaction. The space requirements for checkpoint and disk space availability information are also very small. However, in systems with a high number of transactions that change data, the transaction log can grow to be very large, requiring periodic truncation.

The checkpoint information is deleted at the next checkpoint. Information related to particular savepoints is deleted when the savepoint is released or rolled back.

Concurrency for Backups

Backups may be performed concurrently with read and write operations.

Backup is a DML operation. Backup backs up as of the start of the backup command (the checkpoint). Restore operations, however, require exclusive access, because they write to the database.

Cursors in Transactions

Return the results of a **SELECT** in the form of a data type called a cursor.

A cursor is similar to a table, but one row is identified as the present, or current row. Various commands allow you to navigate through the rows of a cursor. For example, the **FETCH** command retrieves a row from the cursor and identifies it as the current row. You can step through all the rows in a cursor by calling this command repeatedly.

You may find cursors useful when you write procedures, or applications that access a database using Embedded SQL. They are also used by many front-end query tools. They are unavailable when you use Interactive SQL.

SAP Sybase IQ cursors are updatable, which allows you to modify the underlying data in the database while processing a cursor.

The rows in a cursor, like those in a table, have no order associated with them. The **FETCH** command steps through the rows, but the order may appear random and inconsistent. Impose an order by appending an **ORDER BY** phrase to your **SELECT** statement.

The **sp_iqcursorinfo** stored procedure displays information about open cursors on the server.

Cursors and Versioning

When you use cursors, SAP Sybase IQ must be able to manage multiple versions within a single transaction.

For example, assume that you open a cursor called `cust_cursor` at time *x* that uses the `customer` table. You update that table at time *y*. SAP Sybase IQ retains the version of the `customer` table from time x until you are done using `cust_cursor`.

The support of cursors by SAP Sybase IQ is oriented toward their likely use in DSS applications.

Cursor Sensitivity

A cursor is said to be sensitive if its membership—the data rows it returns—can vary from the time it is opened until the time it is closed. An insensitive cursor has its membership fixed when it is opened.

The membership and values of the result set of an insensitive cursor are indeterminate with respect to changes. A value-sensitive cursor is insensitive with respect to its membership and sensitive with respect to the order and values of the result set. SAP Sybase IQ supports insensitive updatable cursors.

Cursor Scrolling

SAP Sybase IQ cursors can be either scrolling or non-scrolling.

Non-scrolling cursors allow only the command forms **FETCH NEXT** and **FETCH RELATIVE 0** to find and retrieve data. They do not keep track of which rows have been fetched. A cursor declared as **DYNAMIC SCROLL** is the same as a cursor declared as **SCROLL**.

Set the option **FORCE_NO_SCROLL_CURSORS** on to save on temporary storage requirements 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 is faster with this option off.

If your front-end application rarely performs backward scrolling, make **FORCE_NO_SCROLL_CURSORS = 'ON'** a permanent **PUBLIC** option. It uses less memory and improves query performance.

Hold Cursors

Specifying the **HOLD** option when you open a cursor keeps the cursor open past the end of the transaction, if the transaction ends in a **COMMIT**.

A hold cursor does not remain open across a **ROLLBACK** in which a cursor is opened.

In SAP Sybase IQ, hold cursors can be updated until they are committed. After it is committed, a hold cursor is marked internally as read-only and subsequent positioned updates generate an error.

Although the **HOLD** option is not commonly used in a DSS environment, it may prove useful in some situations. For example, many existing applications expect to use hold cursors, and some ODBC drivers use hold cursors by default.

SAP Sybase IQ provides the version management needed for hold cursors.

Hold cursors do impact performance. All resources used by the cursor, including memory, disk space, and process threads, are held until the cursor is closed.

Positioned Operations

In a positioned operation, the current location of the cursor determines where a read or write operation begins.

SAP Sybase IQ supports positioned fetches, which can be helpful in long query transactions. SAP Sybase IQ also supports positioned update and delete operations, which are intended for shorter insertions and deletions. For the most part, updates to SAP Sybase IQ databases are likely to involve large amounts of data; repositioning is a very minor part of such write operations.

Positioned updates and deletes are handled as operations on the cursor, and therefore part of its transaction, rather than as separate statements. Any failure that occurs after the cursor is open results in a rollback of all operations that have been performed through this open cursor.

Message Logging for Cursors

By default, cursor operations are not logged in the SAP Sybase IQ message file.

To track cursor operations to determine the cause of a problem, turn on the `LOG_CURSOR_OPERATIONS` option to produce a message each time a cursor is opened or closed. Data changes made through an updatable cursor are also logged in the IQ message file.

Remote Transactions

Transaction management involving remote servers uses a *two-phase commit* protocol.

SAP Sybase IQ implements a strategy that ensures transaction integrity for most scenarios.

Remote Transaction Restrictions

Remote transaction management has savepoints and nested statement restrictions.

Restrictions on transaction management include:

- Savepoints are not propagated to remote servers.
- If nested **BEGIN TRANSACTION** and **COMMIT TRANSACTION** statements are included in a transaction that involves remote servers, only the outermost set of statements is processed. The inconsistent set, containing the **BEGIN TRANSACTION** and **COMMIT TRANSACTION** statements, is not transmitted to remote servers.

Create Procedures and Batches

Procedures and batches enhance the security, efficiency, and standardization of SAP Sybase IQ databases.

Procedures store SQL statements in the database for use by all applications. They enhance the security, efficiency, and standardization of databases. User-defined functions are a type of procedure that return a value to the calling environment for use in queries and other SQL statements.

For many purposes, server-side JDBC provides a more flexible way to build logic into the database than SQL stored procedures.

Batches are sets of SQL statements that are submitted to the database server as a group. Many features available in procedures, such as control statements, are also available in batches.

Note: Use source control software to track changes to source code, and changes to objects created from source (including stored procedures), that you deploy to the database.

Procedures

Creating Procedures

To create an SAP Sybase IQ procedure, use the SQL statement **CREATE PROCEDURE**.

Prerequisites

You must have the CREATE PROCEDURE or CREATE ANY PROCEDURE system privilege

Task

Enter a **CREATE PROCEDURE** statement.

This example uses the SAP Sybase IQ demo database `iqdemo.db`.

```
CREATE PROCEDURE new_dept(IN id INT,
             IN name CHAR(35),
                IN head_id INT)
BEGIN
        INSERT
                INTO GROUPO.departments(DepartmentID,
                DepartmentName,
                DepartmentHeadID)
                values (id, name, head_id);
END
```

Note: Use the `AT location-string` SQL syntax of **CREATE PROCEDURE** to create a proxy stored procedure.

Altering Procedures

You can modify an existing procedure using either Sybase Control Center or Interactive SQL. You must have the ALTER ANY PROCEDURE system privilege or be the owner of the procedure.

You can also modify procedures using the **ALTER PROCEDURE** statement.

Calling Procedures

CALL statements invoke procedures. Procedures can be called by an application program or by other procedures.

Deleting Procedures

Once you create a procedure, it remains in the database until someone explicitly removes it. Only the owner of the procedure or a user with the DROP ANY PROCEDURE system privilege can drop the procedure from the database.

Execute a **DROP PROCEDURE** statement to drop the procedure from the database.

Privileges to Execute Stored Procedures

A procedure is owned by the user who created it; that user can execute it without privilege.

Note: This information does not apply to system procedures. See *Reference: Building Blocks, Tables, and Procedures > System Procedures* for details on granting privilege to run system procedures.

Permission to execute the procedure can be granted to other users using the **GRANT EXECUTE** command. For example, the owner of the procedure **new_dept** allows **another_user** to execute the procedure using:

```
GRANT EXECUTE ON new_dept TO another_user
```

To revoke permission to execute the procedure, execute:

```
REVOKE EXECUTE ON new_dept FROM another_user
```

Returning Procedure Results in Parameters

Procedures return results to the calling environment in several ways.

- Individual values are returned as OUT or INOUT parameters.
- As result sets.
- A single result can be returned using a **RETURN** statement.

Note: This example uses the SAP Sybase IQ demo database `iqdemo.db`.

```
CREATE PROCEDURE SalaryList (IN department_id INT)
RESULT ( "Employee ID" INT, "Salary" NUMERIC(20,3) )
BEGIN
            SELECT EmployeeID, Salary
            FROM Employees
            WHERE Employees.DepartmentID = department_id;
END
```

Returning Procedure Results in Result Sets

In addition to returning results to the calling environment in individual parameters, procedures can return information in result sets. A result set is typically the result of a query.

If a procedure dynamically creates and then selects the same temporary table within a stored procedure, you must use the **EXECUTE IMMEDIATE WITH RESULT SET ON** syntax to avoid `Column not found` errors.

For example:

```
CREATE PROCEDURE p1 (IN @t varchar(30))
   BEGIN
      EXECUTE IMMEDIATE
      'SELECT * INTO #resultSet FROM ' || @t;
      EXECUTE IMMEDIATE WITH RESULT SET ON
      'SELECT * FROM #resultSet';
   END
```

See also

- *Procedure Results* on page 280

Displaying Procedure Information

Use **sp_iqprocedure** to show information about system and user-defined procedures in a database.

In Interactive SQL, run **sp_iqprocedure**.

Displays information about the user-defined procedure `sp_test`:

```
sp_iqprocedure sp_test

proc_name    proc_owner    proc_defn       replicate     srvid     r
emarks

sp_test      DBA       create procedure   N           (NULL)    (NULL)
                         DBA.sp_test(in n1
                         integer)
                         begin message'sp_test'end
```

Displaying Procedure Parameter Information

Use **sp_procparm** to show information about stored procedure parameters.

In Interactive SQL, run **sp_iqprocparm**.

Display information about the parameters of the system procedure **sp_iqshowcompression**:

```
sp_iqprocparm sp_iqshowcompression, dbo, system

proc_name            proc_owner  parm_name    parm_type  parm_mode
domain_name  width  scale  default

sp_iqshowcompression  dbo         @owner_name  normal     in
char         128    0      (NULL)
sp_iqshowcompression  dbo         @table_name  normal     in
char         128    0      (NULL)
sp_iqshowcompression  dbo         @column_name normal     in
char         128    0      (NULL)
sp_iqshowcompression  dbo         Column       result     out
char         128    0      (NULL)
sp_iqshowcompression  dbo         Compression  result     out
char         3      0      (NULL)
```

Cursors in Procedures

Cursors retrieve rows one at a time from a query or stored procedure that has multiple rows in its result set.

A cursor is a handle or an identifier for the query or procedure, and for a current position within the result set.

Cursor management

Managing a cursor is similar to managing a file in a programming language. The following steps manage cursors:

1. Declare a cursor for a particular SELECT statement or procedure using the DECLARE statement.
2. Open the cursor using the OPEN statement.
3. Use the FETCH statement to retrieve results one row at a time from the cursor.
4. A row not found warning signals the end of the result set.
5. Close the cursor using the CLOSE statement.

By default, cursors are automatically closed at the end of a transaction (on COMMIT or ROLLBACK statements). Cursors opened using the WITH HOLD clause stay open for subsequent transactions until explicitly closed.

Cursor Positioning

Cursor positioning is flexible. When a cursor is opened, it is positioned before the first row. You can move the cursor position to an absolute position from the start or the end of the query results, or to a position relative to the current cursor position. The specifics of how you change cursor position, and what operations are possible, are governed by the programming interface

The number of row positions you can fetch in a cursor is governed by the size of an integer. You can fetch rows numbered up to number 2147483646, which is one less than the value that can

be held in an integer. When using negative numbers (rows from the end) you can fetch down to one more than the largest negative value that can be held in an integer.

You can use special positioned update and delete operations to update or delete the row at the current position of the cursor. If the cursor is positioned before the first row or after the last row, an error is returned indicating that there is no corresponding cursor row.

Note: Inserts and some updates to asensitive cursors can cause problems with cursor positioning. SAP Sybase IQ does not put inserted rows at a predictable position within a cursor unless there is an `ORDER BY` clause on the SELECT statement. Sometimes the inserted row does not appear at all until the cursor is closed and opened again. With SAP Sybase IQ, this occurs if a work table had to be created to open the cursor.

The `UPDATE` statement may cause a row to move in the cursor. This happens if the cursor has an ORDER BY clause that uses an existing index (a work table is not created). Using `STATIC SCROLL` cursors alleviates these problems but requires more memory and processing.

Note: SAP Sybase IQ treats the `FIRST`, `LAST`, and `ABSOLUTE` options as starting from the beginning of the result set. It treats `RELATIVE` as a negative row count as starting from the current position.

Cursors on SELECT statements

The following procedure uses a cursor on a SELECT statement. Based on the same query used in the ListCustomerValue procedure, it illustrates several features of the stored procedure language.

```
CREATE PROCEDURE TopCustomerValue(
      OUT TopCompany CHAR(36),
      OUT TopValue INT )
BEGIN
   -- 1. Declare the "row not found" exception
   DECLARE err_notfound
      EXCEPTION FOR SQLSTATE '02000';
   -- 2.  Declare variables to hold
   --    each company name and its value
   DECLARE ThisName CHAR(36);
   DECLARE ThisValue INT;
   -- 3.  Declare the cursor ThisCompany
   --     for the query
   DECLARE ThisCompany CURSOR FOR
   SELECT CompanyName,
         CAST( sum( SalesOrderItems.Quantity *
               Products.UnitPrice ) AS INTEGER )
         AS value
   FROM Customers
      INNER JOIN SalesOrders
      INNER JOIN SalesOrderItems
      INNER JOIN Products
   GROUP BY CompanyName;
   -- 4. Initialize the values of TopValue
   SET TopValue = 0;
```

```
   -- 5. Open the cursor
   OPEN ThisCompany;
   -- 6. Loop over the rows of the query
   CompanyLoop:
   LOOP
      FETCH NEXT ThisCompany
         INTO ThisName, ThisValue;
      IF SQLSTATE = err_notfound THEN
         LEAVE CompanyLoop;
      END IF;
      IF ThisValue > TopValue THEN
         SET TopCompany = ThisName;
         SET TopValue = ThisValue;
      END IF;
   END LOOP CompanyLoop;
   -- 7. Close the cursor
   CLOSE ThisCompany;
END;
```

Notes

The TopCustomerValue procedure has the following notable features:

- An exception is declared. This exception signals, later in the procedure, when a loop over the results of a query completes.
- Two local variables ThisName and ThisValue are declared to hold the results from each row of the query.
- The cursor ThisCompany is declared. The SELECT statement produces a list of company names and the total value of the orders placed by that company.
- The value of TopValue is set to an initial value of 0, for later use in the loop.
- The ThisCompany cursor opens.
- The LOOP statement loops over each row of the query, placing each company name in turn into the variables ThisName and ThisValue. If ThisValue is greater than the current top value, TopCompany and TopValue are reset to ThisName and ThisValue.
- The cursor closes at the end of the procedure.
- You can also write this procedure without a loop by adding an ORDER BY value DESC clause to the SELECT statement. Then, only the first row of the cursor needs to be fetched.

The LOOP construct in the TopCompanyValue procedure is a standard form, exiting after the last row is processed. You can rewrite this procedure in a more compact form using a FOR loop. The FOR statement combines several aspects of the above procedure into a single statement.

```
CREATE PROCEDURE TopCustomerValue2(
      OUT TopCompany CHAR(36),
      OUT TopValue INT )
BEGIN
   -- 1. Initialize the TopValue variable
   SET TopValue = 0;
   -- 2. Do the For Loop
   FOR CompanyFor AS ThisCompany
      CURSOR FOR
```

```
      SELECT CompanyName AS ThisName,
         CAST( sum( SalesOrderItems.Quantity *
               Products.UnitPrice ) AS INTEGER )
         AS ThisValue
      FROM Customers
         INNER JOIN SalesOrders
         INNER JOIN SalesOrderItems
         INNER JOIN Products
      GROUP BY ThisName
   DO
      IF ThisValue > TopValue THEN
         SET TopCompany = ThisName;
         SET TopValue = ThisValue;
      END IF;
   END FOR;
END;
```

Using IQ UTILITIES to Create Stored Procedures

The system stored procedures provided in SAP Sybase IQ are implemented in SQL.

You must use the local temporary table and **IQ UTILITIES** statement in exactly the same way as system stored procedures:

All SQL code for procedures is encrypted and compiled into the shared library `libiqscripts16_r.so` file on UNIX and `iqscripts16.dll` file on Windows.

Warning! Failing to use the statements correctly can cause serious problems for your IQ server or database.

To view the stored procedures code, enter **sp_helptext 'owner.procname'** in Interactive SQL.

The syntax for **IQ UTILITIES** is:

```
IQ UTILITIES  MAIN INTO local-temp-table-name arguments
```

You may want to create your own variants of procedures. For example:

- Create a procedure that calls a system stored procedure.
- Create a procedure that is independent of the system stored procedures but performs a similar function.
- Create a procedure that uses the same structure as the system stored procedures but provides additional functionality. For example to display procedure results in graphical form in a front-end tool or browser rather than as text.
- If you choose either of the two previous options, make sure you understand the **IQ UTILITIES** statement and the strict requirements for using it.

IQ UTILITIES Command

IQ UTILITIES is the underlying statement that executes whenever you run most IQ system procedures. In most cases, users are unaware that **IQ UTILITIES** is executing. The only time **IQ UTILITIES** is issued directly by users is to run the IQ buffer cache monitor.

IQ UTILITIES provides a systematic way to collect and report on information maintained in the IQ system tables. There is no general user interface; you can use **IQ UTILITIES** only in the ways that existing system procedures do.

System procedures declare local temporary tables in which to store information. When you execute a system procedure, it in turn executes**IQ UTILITIES** to get the information from the system tables and store it in the local temporary table. The system procedures may simply report the information from the local temporary table or perform additional processing.

In some system procedures, the **IQ UTILITIES** statement includes a predefined number as one of its arguments. This number performs a specific function, for example, deriving a value from information in the system tables.

Choosing Procedures to Call

You can safely use **IQ UTILITIES** to create your own versions of documented system procedures that report on information in the database.

For example, **sp_iqspaceused** displays information about used and available space available in the IQ main and IQ temporary stores. Verify that the procedure you create from a system stored procedure has the correct owner.

Do not create your own versions of system procedures that control IQ operations. Modifying procedures that control IQ operations can lead to serious problems.

Numbers Used by IQ UTILITIES

Numbers are used as arguments in the **IQ UTILITIES** command. Each number is used in conjucntion with a system procedure.

Table 22. IQ UTILITIES values used in system procedures

Number	Procedure	Comments
10000	**sp_iqtransaction**	
20000	**sp_iqconnection** and **sp_iqmpx-countdbremote**	
30000	**sp_iqspaceused**	
40000	**sp_iqspaceinfo**	
50000	**sp_iqlocks**	
60000	**sp_iqmpxversionfetch**	Do Not Use

Number	Procedure	Comments
70000	**sp_iqmpxdumptlvlog**	
80000	**sp_iqcontext**	
100000	**sp_iqindexfragmentation**	
110000	**sp_iqrowdensity**	

Procedure Testing

To maintain the stability of your IQ server and database, test your procedures in a development environment before you run them in a production environment.

User-Defined Functions

User-defined functions are a class of procedures that return a single value to the calling environment.

Creating User-Defined Functions

Use the **CREATE FUNCTION** statement to create user-defined functions.

Prerequisites

- You must have the CREATE PROCEDURE system privilege to create functions owned by you.
- You must have the CREATE ANY PROCEDURE or CREATE ANY OBJECT system privilege to create functions owned by others.
- If the procedure contains an external reference, you must have the CREATE EXTERNAL REFERENCE system privilege, in addition to the above system privileges.

To create a user-defined function in Sybase Control Center, see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

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);
  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.

Calling User-Defined Functions

You can use user-defined functions, subject to permissions, in the same places you would use built-in nonaggregate functions.

This Interactive SQL statement returns a full name from two columns containing a first and last name:

```
SELECT fullname (GivenName, LastName)
FROM Employees;
```

fullname (Employees.GivenName,Employees.SurName)
Fran Whitney
Matthew Cobb
Philip Chin
...

The following statement returns a full name from a supplied first and last name:

```
SELECT fullname ('Jane', 'Smith');
```

fullname ('Jane','Smith')
Jane Smith

Any user who has been granted Execute permissions for the function can use the *fullname* function.

Dropping a user-defined function (SQL)

User-defined functions remain in the database until they are explicitly removed.

Prerequisites

You must be the owner of the user-defined function or have one of the following system privileges:

DROP ANY PROCEDURE
DROP ANY OBJECT

Task

1. Connect to the database.
2. Execute a DROP FUNCTION statement similar to the following:
   ```
   DROP FUNCTION function-name;
   ```

The user-defined function is dropped.

Permissions to Execute User-Defined Functions

A user-defined function is owned by the user who created it; that user can execute it without permission.

The owner of a user-defined function can grant permissions to other users with the **GRANT EXECUTE** command.

For example, the creator of the function `fullname` allows `another_user` to use `fullname` the function using:

```
GRANT EXECUTE ON fullname TO another_user
```

This statement revokes permission to use the function:

```
REVOKE EXECUTE ON fullname FROM another_user
```

See *Administration: User Management and Security > Security Management > Privileges > Object-Level Privileges > Grant and Revoke Object Level Privileges > Granting the EXECUTE Privilege on Functions and Procedures*.

Granting the ability to execute a user-defined function (SQL)

Grant the ability to execute a user-defined function by granting the EXECUTE object-level privilege.

Prerequisites

You must be the owner of the user-defined function, or have EXECUTE privilege with administrative rights on the function.

Ownership of a user-defined function belongs to the user who created it, and no privilege is required for that user to execute it.

Task

You have created a function and you want other user to be able to use it.

1. Connect to the database.
2. Execute a GRANT EXECUTE statement similar to the following:
   ```
   GRANT EXECUTE ON function-name TO user-id;
   ```

The grantee can now execute the procedure.

Batches

A batch is a set of SQL statements submitted together and executed as a group, one after the other. The control statements used in procedures (CASE, IF, LOOP, and so on) can also be

used in batches. If the batch consists of a compound statement enclosed in a BEGIN/END, then it can also contain host variables, local declarations for variables, cursors, temporary tables and exceptions. Host variable references are permitted within batches with the following restrictions:

- only one statement in the batch can refer to host variables
- the statement which uses host variables cannot be preceded by a statement which returns a result set

Use of BEGIN/END is recommended to clearly indicate when a batch is being used.

Statements within the batch may be delimited with semicolons, in which case the batch is conforming to the Watcom SQL dialect. A multi-statement batch that does not use semicolons to delimit statements conforms to the Transact-SQL dialect. The dialect of the batch determines which statements are permitted within the batch, and also determines how errors within the batch are handled.

In many ways, batches are similar to stored procedures; however, there are some differences:

- batches do not have names
- batches do not accept parameters
- batches are not stored persistently in the database
- batches cannot be shared by different connections

A simple batch consists of a set of SQL statements with no delimiters followed by a separate line with just the word go on it. The following example creates an Eastern Sales department and transfers all sales reps from Massachusetts to that department. It is an example of a Transact-SQL batch.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' )

UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA'

COMMIT
go
```

The word go is recognized by Interactive SQL and causes it to send the previous statements as a single batch to the server.

The following example, while similar in appearance, is handled quite differently by Interactive SQL. This example does not use the Transact-SQL dialect. Each statement is delimited by a semicolon. Interactive SQL sends each semicolon-delimited statement separately to the server. It is not treated as a batch.

```
INSERT
INTO Departments ( DepartmentID, DepartmentName )
VALUES ( 220, 'Eastern Sales' );
```

```
UPDATE Employees
SET DepartmentID = 220
WHERE DepartmentID = 200
AND State = 'MA';

COMMIT;
```

To have Interactive SQL treat it as a batch, it can be changed into a compound statement using `BEGIN ... END`. The following is a revised version of the previous example. The three statements in the compound statement are sent as a batch to the server.

```
BEGIN
  INSERT
  INTO Departments ( DepartmentID, DepartmentName )
  VALUES ( 220, 'Eastern Sales' );

  UPDATE Employees
  SET DepartmentID = 220
  WHERE DepartmentID = 200
  AND State = 'MA';

  COMMIT;
END
```

In this particular example, it makes no difference to the end result whether a batch or individual statements are executed by the server. There are situations, though, where it can make a difference. Consider the following example.

```
DECLARE @CurrentID INTEGER;
SET @CurrentID = 207;
SELECT Surname FROM Employees
  WHERE EmployeeID=@CurrentID;
```

If you execute this example using Interactive SQL, the database server returns an error indicating that the variable cannot be found. This happens because Interactive SQL sends three separate statements to the server. They are not executed as a batch. As you have already seen, the remedy is to use a compound statement to force Interactive SQL to send these statements as a batch to the server. The following example accomplishes this.

```
BEGIN
  DECLARE @CurrentID INTEGER;
  SET @CurrentID = 207;
  SELECT Surname FROM Employees
    WHERE EmployeeID=@CurrentID;
END
```

Putting a BEGIN and END around a set of statements forces Interactive SQL to treat them as a batch.

The IF statement is another example of a compound statement. Interactive SQL sends the following statements as a single batch to the server.

```
IF EXISTS(   SELECT *
             FROM SYSTAB
```

```
            WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
ELSE
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF
```

This situation does not arise when using other techniques to prepare and execute SQL statements. For example, an application that uses ODBC can prepare and execute a series of semicolon-separated statements as a batch.

Care must be exercised when mixing Interactive SQL statements with SQL statements intended for the server. The following is an example of how mixing Interactive SQL statements and SQL statements can be an issue. In this example, since the Interactive SQL OUTPUT statement is embedded in the compound statement, it is sent along with all the other statements to the server as a batch, and results in a syntax error.

```
IF EXISTS(   SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
   OUTPUT TO 'c:\\temp\\query.txt';
ELSE
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF
```

The correct placement of the OUTPUT statement is shown below.

```
IF EXISTS(   SELECT *
            FROM SYSTAB
            WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
ELSE
```

```
   MESSAGE 'The Employees table does not exist'
   TO CLIENT;
END IF;
OUTPUT TO 'c:\\temp\\query.txt';
```

Control statements

There are several control statements for logical flow and decision making in the body of a procedure, trigger, or user-defined function, or in a batch. Available control statements include:

Control statement	Syntax

Compound statements	`BEGIN [ ATOMIC ]`
	`   Statement-list`
	`END`

Conditional execution: IF	`IF condition THEN`
	`   Statement-list`
	`ELSEIF condition THEN`
	`   Statement-list`
	`ELSE`
	`   Statement-list`
	`END IF`

Conditional execution: CASE	`CASE expression`
	`WHEN value THEN`
	`   Statement-list`
	`WHEN value THEN`
	`   Statement-list`
	`ELSE`
	`   Statement-list`
	`END CASE`

Repetition: WHILE, LOOP	`WHILE condition LOOP`
	`   Statement-list`
	`END LOOP`

Repetition: FOR cursor loop	`FOR loop-name`
	`   AS cursor-name CURSOR FOR`
	`   select-statement`
	`DO`
	`   Statement-list`
	`END FOR`

Break: LEAVE	`LEAVE label`
CALL	`CALL procname( arg, ... )`

Compound statements

A compound statement starts with the keyword BEGIN and concludes with the keyword END. The body of a procedure or trigger is a **compound statement**. Compound statements can also be used in batches. Compound statements can be nested, and combined with other control statements to define execution flow in procedures and triggers or in batches.

A compound statement allows a set of SQL statements to be grouped together and treated as a unit. Delimit SQL statements within a compound statement with semicolons.

Declarations in compound statements

Local declarations in a compound statement immediately follow the BEGIN keyword. These local declarations exist only within the compound statement. Within a compound statement you can declare:

- Variables
- Cursors
- Temporary tables
- Exceptions (error identifiers)

Local declarations can be referenced by any statement in that compound statement, or in any compound statement nested within it. Local declarations are not visible to other procedures called from the compound statement.

Atomic compound statements

An **atomic** statement is a statement that is 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 changed rows revert back to their original state. The UPDATE statement is atomic.

All non-compound SQL statements are atomic. You can make a compound statement atomic by adding the keyword ATOMIC after the BEGIN keyword.

```
BEGIN ATOMIC
   UPDATE Employees
   SET ManagerID = 501
   WHERE EmployeeID = 467;
    UPDATE Employees
   SET BirthDate = 'bad_data';
END
```

In this example, the two update statements are part of an atomic compound statement. They must either succeed or fail as one. The first update statement would succeed. The second one causes a data conversion error since the value being assigned to the BirthDate column cannot be converted to a date.

The atomic compound statement fails and the effect of both UPDATE statements is undone. Even if the currently executing transaction is eventually committed, neither statement in the atomic compound statement takes effect.

If an atomic compound statement succeeds, the changes made within the compound statement take effect only if the currently executing transaction is committed. In the case when an atomic compound statement succeeds but the transaction in which it occurs gets rolled back, the atomic compound statement also gets rolled back. A savepoint is established at the start of the atomic compound statement. Any errors within the statement result in a rollback to that savepoint.

When an atomic compound statement is executed in autocommit (unchained) mode, the commit mode changes to manual (chained) until statement execution is complete. In manual mode, DML statements executed within the atomic compound statement do not cause an immediate COMMIT or ROLLBACK. If the atomic compound statement completes successfully, a COMMIT statement is executed; otherwise, a ROLLBACK statement is executed.

You cannot use COMMIT and ROLLBACK and some ROLLBACK TO SAVEPOINT statements within an atomic compound statement.

Structure of Procedures

The body of a procedure consists of a compound statement.

A compound statement consists of a **BEGIN** and an **END**, enclosing a set of SQL statements. Semicolons delimit each statement.

SQL Statements Allowed in Procedures

You can use almost all SQL statements within procedures:

- **SELECT**, **UPDATE**, **DELETE**, **INSERT**, and **SET VARIABLE**
- The **CALL** statement to execute other procedures
- Control statements
- Cursor statements
- Exception handling statements
- The **EXECUTE IMMEDIATE** statement

Some SQL statements you cannot use within procedures:

- **CONNECT** statement
- **DISCONNECT** statement

You can use **COMMIT**, **ROLLBACK**, and **SAVEPOINT** statements within procedures with certain restrictions.

See the *Usage* section for each statement in *Reference: Statements and Options > SQL Statements*.

Parameter declaration for procedures

Procedure parameters appear as a list in the CREATE PROCEDURE statement. Parameter names must conform to the rules for other database identifiers such as column names. They must have valid data types, and can be prefixed with one of the keywords IN, OUT or INOUT. By default, parameters are INOUT parameters. These keywords have the following meanings:

- **IN –** The argument is an expression that provides a value to the procedure.
- **OUT –** The argument is a variable that could be given a value by the procedure.
- **INOUT –** The argument is a variable that provides a value to the procedure, and could be given a new value by the procedure.

You can assign default values to procedure parameters in the CREATE PROCEDURE statement. The default value must be a constant, which may be NULL. For example, the following procedure uses the NULL default for an IN parameter to avoid executing a query that would have no meaning:

```
CREATE PROCEDURE CustomerProducts(
      IN customer_ID
                  INTEGER DEFAULT NULL )
RESULT ( product_ID INTEGER,
         quantity_ordered INTEGER )
BEGIN
   IF customer_ID IS NULL THEN
      RETURN;
   ELSE
      SELECT    Products.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;
```

The following statement assigns the DEFAULT NULL, and the procedure RETURNs instead of executing the query.

```
CALL CustomerProducts();
```

Ways to pass parameters to procedures

You can take advantage of default values of stored procedure parameters with either of two forms of the CALL statement.

If the optional parameters are at the end of the argument list in the CREATE PROCEDURE statement, they may be omitted from the CALL statement. As an example, consider a procedure with three INOUT parameters:

```
CREATE PROCEDURE SampleProcedure(
       INOUT var1 INT DEFAULT 1,
                  INOUT var2 int DEFAULT 2,
                  INOUT var3 int DEFAULT 3 )
...
```

This example assumes that the calling environment has set up three variables to hold the values passed to the procedure:

```
CREATE VARIABLE V1 INT;
CREATE VARIABLE V2 INT;
CREATE VARIABLE V3 INT;
```

The procedure SampleProcedure may be called supplying only the first parameter as follows, in which case the default values are used for *var2* and *var3.*

```
CALL SampleProcedure( V1 );
```

The procedure can also be called by providing only the second parameter by using the DEFAULT value for the first parameter, as follows:

```
CALL SampleProcedure( DEFAULT, V2 );
```

A more flexible method of calling procedures with optional arguments is to pass the parameters by name. The SampleProcedure procedure may be called as follows:

```
CALL SampleProcedure( var1 = V1, var3 = V3 );
```

or as follows:

```
CALL SampleProcedure( var3 = V3, var1 = V1 );
```

How to pass parameters to functions

User-defined functions are not invoked with the CALL statement, but are used in the same manner that built-in functions are. For example, the following statement uses the FullName function to retrieve the names of employees:

Notes

- Default parameters can be used in calling functions. However, parameters cannot be passed to functions by name.
- Parameters are passed by value, not by reference. Even if the function changes the value of the parameter, this change is not returned to the calling environment.
- Output parameters cannot be used in user-defined functions.

- User-defined functions cannot return result sets.

Example: List the names of all employees

In Interactive SQL, execute the following query:

```
SELECT FullName( GivenName, Surname ) AS Name
   FROM Employees;
```

The following results appear:

Name
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
...

Procedure Results

Procedures can return either single or multiple rows of data.

Results consisting of a single row of data can be passed back as arguments to the procedure. Results consisting of multiple rows of data are passed back as result sets. Procedures can also return a single value given in the **RETURN** statement.

Returning a value using the RETURN statement

The RETURN statement returns a single integer value to the calling environment, causing an immediate exit from the procedure.

Prerequisites

There are no prerequisites for this task.

Task

1. Execute the following statement:

   ```
   RETURN expression
   ```

2. The value of the supplied expression is returned to the calling environment. Use an extension of the CALL statement to save the return value in a variable:

   ```
   CREATE VARIABLE returnval INTEGER;
   returnval = CALL variable/procedure-name? myproc();
   ```

A value is returned and saved as a variable.

Ways to return results as procedure parameters

Procedures can return results to the calling environment in the parameters to the procedure. Within a procedure, parameters and variables can be assigned values using:

- the SET statement

  The following procedure returns a value in an OUT parameter assigned using a SET statement. You must have the CREATE PROCEDURE system privilege to execute the following statement:

  ```
  CREATE PROCEDURE greater(
     IN a INT,
     IN b INT,
     OUT c INT )
  BEGIN
     IF a > b THEN
        SET c = a;
     ELSE
        SET c = b;
     END IF ;
  END;
  ```

- a SELECT statement with an INTO clause

  A single-row query retrieves at most one row from the database. This type of query uses a SELECT statement with an INTO clause. The INTO clause follows the SELECT list and precedes the FROM clause. It contains a list of variables to receive the value for each SELECT list item. There must be the same number of variables as there are SELECT list items.

  When a SELECT statement executes, the database server retrieves the results of the SELECT statement and places the results in the variables. If the query results contain more than one row, the database server returns an error. For queries returning more than one row, you must use cursors.

  If the query results in no rows being selected, the variables are not updated, and a warning is returned.

  You must have the appropriate SELECT privileges on the object to execute a SELECT statement.

Example 1: Create a procedure and select its results using a SELECT...INTO statement

1. Start Interactive SQL and connect to the SAP Sybase IQ sample database. You must have the CREATE PROCEDURE system privilege and either SELECT privilege on the Employees table or the SELECT ANY TABLE system privilege.
2. In the **SQL Statements** pane, execute the following statement to create a procedure (AverageSalary) that returns the average salary of employees as an OUT parameter:

   ```
   CREATE PROCEDURE AverageSalary( OUT average_salary NUMERIC(20,3) )
   BEGIN
      SELECT AVG( Salary )
      INTO average_salary
   ```

```
   FROM GROUPO.Employees;
END;
```

3. Create a variable to hold the procedure output. In this case, the output variable is numeric, with three decimal places.

```
CREATE VARIABLE Average NUMERIC(20,3);
```

4. Call the procedure using the created variable to hold the result:

```
CALL AverageSalary( Average );
```

5. If the procedure was created and run properly, the Interactive SQL **Messages** tab does not display any errors.
6. To inspect the value of the variable, execute the following statement:

```
SELECT Average;
```

7. Look at the value of the output variable Average. The **Results** tab in the **Results** pane displays the value 49988.623 for this variable, the average employee salary.

Example 2: Returning the results of a single-row SELECT statement

1. Start Interactive SQL and connect to the SAP Sybase IQ sample database. You must have the CREATE PROCEDURE system privilege and either SELECT privilege on the Customers table or the SELECT ANY TABLE system privilege.
2. Execute the following statement to return the number of orders placed by a given customer:

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

3. Test this procedure using the following statements, which show the number of orders placed by the customer with ID 102:

```
CREATE VARIABLE orders INT;
CALL OrderCount ( 102, orders );
SELECT orders;
```

Notes for Example 2 –

- The customer_ID parameter is declared as an IN parameter. This parameter holds the customer ID passed in to the procedure.
- The Orders parameter is declared as an OUT parameter. It holds the value of the orders variable returned to the calling environment.
- No DECLARE statement is necessary for the Orders variable as it is declared in the procedure argument list.
- The SELECT statement returns a single row and places it into the variable Orders.

Information returned in result sets from procedures

In addition to returning results to the calling environment in individual parameters, procedures can return information in result sets. A result set is typically the result of a query.

The number of variables in the RESULT clause must match the number of the SELECT list items. Automatic data type conversion is performed where possible if data types do not match.

The RESULT clause is part of the CREATE PROCEDURE statement, and does not have a statement delimiter.

The names of the SELECT list items do not need to match those in the RESULT clause.

To modify procedure result sets on a view, the user must have the appropriate privileges on the underlying table.

In the case that a stored procedure or user-defined function returns a result, it cannot also support output parameters or return values.

Interactive SQL displays only the first result set by default. To allow a procedure to return more than one row of results in Interactive SQL, set the **Show Multiple Result Sets** option on the **Results** tab of the **Options** window.

Example 1

The following procedure returns a list of customers who have placed orders, together with the total value of the orders placed.

Execute the following statement in Interactive SQL:

```
CREATE PROCEDURE ListCustomerValue()
RESULT ( "Company" CHAR(36), "Value" INT )
BEGIN
   SELECT CompanyName,
      CAST( SUM(  SalesOrderItems.Quantity *
                  Products.UnitPrice )
                  AS INTEGER ) AS value
   FROM Customers
      INNER JOIN SalesOrders
      INNER JOIN SalesOrderItems
      INNER JOIN Products
   GROUP BY CompanyName
   ORDER BY value DESC;
END;
```

Executing `CALL ListCustomerValue ( );` returns the following result set:

Company	Value
The Hat Company	5016
The Igloo	3564

Company	Value
The Ultimate	3348
North Land Trading	3144
Molly's	2808
...	...

Example 2

The following procedure returns a result set containing the salary for each employee in a given department. Execute the following statement in Interactive SQL:

```
CREATE PROCEDURE SalaryList( IN department_id INT )
RESULT ( "Employee ID" INT, Salary NUMERIC(20,3) )
BEGIN
   SELECT EmployeeID, Salary
    FROM Employees
    WHERE Employees.DepartmentID = department_id;
END;
```

The names in the RESULT clause are matched to the results of the query and used as column headings in the displayed results.

To list the salaries of employees in the R & D department (department ID 100), execute the following statement:

```
CALL SalaryList( 100 );
```

The following result set appears in the **Results** pane:

Employee ID	Salary
102	45700.000
105	62000.000
160	57490.000
243	72995.000
...	...

Returning multiple result sets

You can use Interactive SQL to return more than one result set from a procedure.

Prerequisites

There are no prerequisites for this task.

Task

By default, Interactive SQL does not show multiple result sets.

1. In Interactive SQL, connect to the database.
2. Click **Tools** » **Options**.
3. Click **SAP Sybase IQ**.
4. On the **Results** tab, click **Show All Result Sets**.
5. Click **OK**.

After you enable this option, Interactive SQL shows multiple result sets. The setting takes effect immediately and remains in effect for future sessions until it is disabled.

Next

If a RESULT clause is included in a procedure definition, the result sets must be compatible: they must have the same number of items in the SELECT lists, and the data types must all be of types that can be automatically converted to the data types listed in the RESULT clause.

If the RESULT clause is omitted, a procedure can return result sets that vary in the number and type of columns that are returned.

Variable result sets for procedures

The RESULT clause is optional in procedures. Omitting the result clause allows you to write procedures that return different result sets, with different numbers or types of columns, depending on how they are executed.

If you do not use the variable result sets feature, you should use a RESULT clause for performance reasons.

For example, the following procedure returns two columns if the input variable is Y, but only one column otherwise:

```
CREATE PROCEDURE Names( IN formal char(1) )
BEGIN
   IF formal = 'y' THEN
      SELECT Surname, GivenName
      FROM Employees
   ELSE
      SELECT GivenName
      FROM Employees
   END IF
END;
```

The use of variable result sets in procedures is subject to some limitations, depending on the interface used by the client application.

- **Embedded SQL –** To get the proper shape of the result set, you must DESCRIBE the procedure call after the cursor for the result set is opened, but before any rows are returned.

When you create a procedure without a RESULT clause and the procedure returns a variable result set, a DESCRIBE of a SELECT statement that references the procedure may fail. To prevent the failure of the DESCRIBE, it is recommended that you include a WITH clause in the FROM clause of the SELECT statement. Alternately, you could use the WITH VARIABLE RESULT clause in the DESCRIBE statement. The WITH VARIABLE RESULT clause can be used to determine if the procedure call should be described following each OPEN statement.

- **ODBC –** Variable result set procedures can be used by ODBC applications. The SAP Sybase IQ ODBC driver performs the proper description of the variable result sets.
- **Open Client applications –** Open Client applications can use variable result set procedures. SAP Sybase IQ performs the proper description of the variable result sets.

Error and warning handling

After an application program executes a SQL statement, it can examine a **status code**. This status code (or return code) indicates whether the statement executed successfully or failed and gives the reason for the failure. You can use the same mechanism to indicate the success or failure of a CALL statement to a procedure.

Error reporting uses either the SQLCODE or SQLSTATE status descriptions.

Whenever a SQL statement executes, a value appears in special procedure variables called SQLSTATE and SQLCODE. The special value indicates whether there were any unusual conditions encountered when the statement was executed. You can check the value of SQLSTATE or SQLCODE in an IF statement following a SQL statement, and take actions depending on whether the statement succeeded or failed.

For example, the SQLSTATE variable can be used to indicate if a row is successfully fetched. The TopCustomerValue procedure used the SQLSTATE test to detect that all rows of a SELECT statement had been processed.

Default handling of errors

This section describes how SAP Sybase IQ handles errors that occur during a procedure execution, if you have no error handling built in to the procedure.

For different behavior, you can use exception handlers.

Warnings are handled in a slightly different manner from errors.

There are two ways of handling errors without using explicit error handling:

- **Default error handling –** The procedure or trigger fails and returns an error code to the calling environment.
- **ON EXCEPTION RESUME –** If the ON EXCEPTION RESUME clause appears in the CREATE PROCEDURE statement, the procedure carries on executing after an error, resuming at the statement following the one causing the error.

The precise behavior for procedures that use ON EXCEPTION RESUME is dictated by the on_tsql_error option setting.

Default error handling

Generally, if a SQL statement in a procedure or trigger fails, the procedure or trigger stops executing and control returns to the application program with an appropriate setting for the SQLSTATE and SQLCODE values. This is true even if the error occurred in a procedure or trigger invoked directly or indirectly from the first one. For triggers the operation causing the trigger is also undone and the error is returned to the application.

The following demonstration procedures show what happens when an application calls the procedure OuterProc, and OuterProc in turn calls the procedure InnerProc, which then encounters an error.

```
CREATE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc.' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ',
      SQLSTATE,' in OuterProc.' TO CLIENT
END;
CREATE PROCEDURE InnerProc()
   BEGIN
      DECLARE column_not_found
         EXCEPTION FOR SQLSTATE '52003';
      MESSAGE 'Hello from InnerProc.' TO CLIENT;
      SIGNAL column_not_found;
   MESSAGE 'SQLSTATE set to ',
      SQLSTATE, ' in InnerProc.' TO CLIENT;
END;

CALL OuterProc();
```

The Interactive SQL Messages tab displays the following:

```
Hello from OuterProc.
Hello from InnerProc.
```

The DECLARE statement in InnerProc declares a symbolic name for one of the predefined SQLSTATE values associated with error conditions already known to the server.

The MESSAGE statement sends a message to the Interactive SQL Messages tab.

The SIGNAL statement generates an error condition from within the InnerProc procedure.

None of the statements following the SIGNAL statement in InnerProc execute: InnerProc immediately passes control back to the calling environment, which in this case is the procedure OuterProc. None of the statements following the CALL statement in OuterProc execute. The error condition returns to the calling environment to be handled there. For example, Interactive SQL handles the error by displaying a message window describing the error.

The TRACEBACK function provides a list of the statements that were executing when the error occurred. You can use the TRACEBACK function from Interactive SQL by entering the following statement:

```
SELECT TRACEBACK();
```

Error handling with ON EXCEPTION RESUME

If the ON EXCEPTION RESUME clause appears in the CREATE PROCEDURE statement, the procedure checks the following statement when an error occurs. If the statement handles the error, then the procedure continues executing, resuming at the statement after the one causing the error. It does not return control to the calling environment when an error occurred.

The behavior for procedures that use ON EXCEPTION RESUME can be modified by the on_tsql_error option setting.

Error-handling statements include the following:

IF
SELECT @variable =
CASE
LOOP
LEAVE
CONTINUE
CALL
EXECUTE
SIGNAL
RESIGNAL
DECLARE
SET VARIABLE

The following demonstration procedures show what happens when an application calls the procedure OuterProc; and OuterProc in turn calls the procedure InnerProc, which then encounters an error. These demonstration procedures are based on those used earlier in this section:

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE OuterProc()
ON EXCEPTION RESUME
BEGIN
   DECLARE res CHAR(5);
   MESSAGE 'Hello from OuterProc.' TO CLIENT;
   CALL InnerProc();
   SET res=SQLSTATE;
   IF res='52003' THEN
      MESSAGE 'SQLSTATE set to ',
         res, ' in OuterProc.' TO CLIENT;
   END IF
```

```
END;

CREATE PROCEDURE InnerProc()
ON EXCEPTION RESUME
BEGIN
   DECLARE column_not_found
      EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from InnerProc.' TO CLIENT;
   SIGNAL column_not_found;
   MESSAGE 'SQLSTATE set to ',
   SQLSTATE, ' in InnerProc.' TO CLIENT;
END;

CALL OuterProc();
```

The Interactive SQL Messages tab then displays the following:

```
Hello from OuterProc.
Hello from InnerProc.
SQLSTATE set to 52003 in OuterProc.
```

The execution path taken is as follows:

1. OuterProc executes and calls InnerProc.
2. In InnerProc, the SIGNAL statement signals an error.
3. The MESSAGE statement is not an error-handling statement, so control is passed back to OuterProc and the message is not displayed.
4. In OuterProc, the statement following the error assigns the SQLSTATE value to the variable named `res`. This is an error-handling statement, and so execution continues and the OuterProc message appears.

Default handling of warnings

Errors and warnings are handled differently. While the default action for errors is to set a value for the SQLSTATE and SQLCODE variables, and return control to the calling environment in the event of an error, the default action for warnings is to set the SQLSTATE and SQLCODE values and continue execution of the procedure.

The following demonstration procedures illustrate default handling of warnings.

In this case, the SIGNAL statement generates a condition indicating that the row cannot be found. This is a warning rather than an error.

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc.' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ',
      SQLSTATE,' in OuterProc.' TO CLIENT;
END;
CREATE PROCEDURE InnerProc()
BEGIN
```

```
   DECLARE row_not_found
      EXCEPTION FOR SQLSTATE '02000';
   MESSAGE 'Hello from InnerProc.' TO CLIENT;
   SIGNAL row_not_found;
   MESSAGE 'SQLSTATE set to ',
   SQLSTATE, ' in InnerProc.' TO CLIENT;
END;

CALL OuterProc();
```

The Interactive SQL Messages tab then displays the following:

```
Hello from OuterProc.
Hello from InnerProc.
SQLSTATE set to 02000 in InnerProc.
SQLSTATE set to 00000 in OuterProc.
```

The procedures both continued executing after the warning was generated, with SQLSTATE set by the warning (02000).

Execution of the second MESSAGE statement in InnerProc resets the warning. Successful execution of any SQL statement resets SQLSTATE to 00000 and SQLCODE to 0. If a procedure needs to save the error status, it must do an assignment of the value immediately after execution of the statement which caused the error or warning.

Exception handlers

It is often desirable to intercept certain types of errors and handle them within a procedure or trigger, rather than pass the error back to the calling environment. This is done through the use of an **exception handler**.

You define an exception handler with the EXCEPTION part of a compound statement.

Whenever an error occurs in the compound statement, the exception handler executes. Unlike errors, warnings do not cause exception handling code to be executed. Exception handling code also executes if an error appears in a nested compound statement or in a procedure or trigger invoked anywhere within the compound statement.

An exception handler for the interrupt error SQL_INTERRUPT, SQLSTATE 57014 should only contain non-interruptible statements such as ROLLBACK and ROLLBACK TO SAVEPOINT. If the exception handler contains interruptible statements that are invoked when the connection is interrupted, the database server stops the exception handler at the first interruptible statement and returns the interrupt error.

An exception handler can use the SQLSTATE or SQLCODE special values to determine why a statement failed. Alternatively, the ERRORMSG function can be used without an argument to return the error condition associated with a SQLSTATE. Only the first statement in each WHEN clause can specify this information and the statement cannot be a compound statement.

In this example, additional code handles the error about the column that cannot be found in the InnerProc procedure.

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE OuterProc()
BEGIN
   MESSAGE 'Hello from OuterProc.' TO CLIENT;
   CALL InnerProc();
   MESSAGE 'SQLSTATE set to ',
      SQLSTATE,' in OuterProc.' TO CLIENT
END;
CREATE PROCEDURE InnerProc()
BEGIN
   DECLARE column_not_found
    EXCEPTION FOR SQLSTATE '52003';
   MESSAGE 'Hello from InnerProc.' TO CLIENT;
  SIGNAL column_not_found;
  MESSAGE 'Line following SIGNAL.' TO CLIENT;
   EXCEPTION
      WHEN column_not_found THEN
        MESSAGE 'Column not found handling.' TO CLIENT;
      WHEN OTHERS THEN
        RESIGNAL ;
END;

CALL OuterProc();
```

The Interactive SQL **Messages** tab then displays the following:

```
Hello from OuterProc.
Hello from InnerProc.
Column not found handling.
SQLSTATE set to 00000 in OuterProc.
```

The EXCEPTION clause declares the exception handler. The lines following EXCEPTION do not execute unless an error occurs. Each WHEN clause specifies an exception name (declared with a DECLARE statement) and the statement or statements to be executed in the event of that exception. The WHEN OTHERS THEN clause specifies the statement(s) to be executed when the exception that occurred does not appear in the preceding WHEN clauses.

In the above example, the statement RESIGNAL passes the exception on to a higher-level exception handler. RESIGNAL is the default action if WHEN OTHERS THEN is not specified in an exception handler.

Additional notes

- The EXCEPTION handler executes, rather than the lines following the SIGNAL statement in InnerProc.
- As the error encountered was an error about a column that cannot be found, the MESSAGE statement included to handle the error executes, and SQLSTATE resets to zero (indicating no errors).
- After the exception handling code executes, control passes back to OuterProc, which proceeds as if no error was encountered.

- You should not use ON EXCEPTION RESUME together with explicit exception handling. The exception handling code is not executed if ON EXCEPTION RESUME is included.
- If the error handling code for the error is a RESIGNAL statement, control returns to the OuterProc procedure with SQLSTATE still set at the value 52003. This is just as if there were no error handling code in InnerProc. Since there is no error handling code in OuterProc, the procedure fails.

Exception handling and atomic compound statements

If an error occurs within an atomic compound statement and that statement has an exception handler that handles the error, then the compound statement completes without an active exception and the changes before the exception are not reversed. If the exception handler does not handle the error or causes another error (including via RESIGNAL), then changes made within the atomic statement are undone.

Nested compound statements and exception handlers

The code following a statement that causes an error executes only if an ON EXCEPTION RESUME clause appears in a procedure definition.

You can use nested compound statements to give you more control over which statements execute following an error and which do not.

The following example illustrates how nested compound statements can be used to control flow.

```
DROP PROCEDURE OuterProc;
DROP PROCEDURE InnerProc;

CREATE PROCEDURE InnerProc()
BEGIN
   BEGIN
      DECLARE column_not_found
      EXCEPTION FOR SQLSTATE VALUE '52003';
      MESSAGE 'Hello from InnerProc' TO CLIENT;
      SIGNAL column_not_found;
      MESSAGE 'Line following SIGNAL' TO CLIENT
      EXCEPTION
         WHEN column_not_found THEN
            MESSAGE 'Column not found handling' TO
            CLIENT;
         WHEN OTHERS THEN
            RESIGNAL;
   END;
   MESSAGE 'Outer compound statement' TO CLIENT;
END;

CALL InnerProc();
```

The Interactive SQL **Messages** tab then displays the following:

```
Hello from InnerProc
Column not found handling
Outer compound statement
```

When the SIGNAL statement that causes the error is encountered, control passes to the exception handler for the compound statement, and the `Column not found handling` message prints. Control then passes back to the outer compound statement and the `Outer compound statement` message prints.

If an error other than Column not found (SQLSTATE) is encountered in the inner compound statement, the exception handler executes the RESIGNAL statement. The RESIGNAL statement passes control directly back to the calling environment, and the remainder of the outer compound statement is not executed.

Example

This example shows the output of the sa_error_stack_trace system procedure with RESIGNAL:

```
CREATE PROCEDURE DBA.error_reporting_procedure()
BEGIN
        SELECT *
        FROM sa_error_stack_trace();
END;

CREATE PROCEDURE DBA.proc1()
BEGIN TRY
    BEGIN TRY
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    END TRY
    BEGIN CATCH
        CALL DBA.proc2();
    END CATCH
END TRY
BEGIN CATCH
    CALL DBA.error_reporting_procedure();
END CATCH;

CREATE PROCEDURE DBA.proc2()
BEGIN
    CALL DBA.proc3();
END;

CREATE PROCEDURE DBA.proc3()
BEGIN
    RESIGNAL;
END;
```

When the procedure above is invoked using CALL proc1(), the following result set is produced:

StackLevel	UserName	ProcName	LineNumber	IsResignal
1	DBA	proc1	8	0
2	DBA	proc2	3	0
3	DBA	proc3	3	1
4	DBA	proc1	5	0

This example shows the output of the sa_error_stack_trace system procedure with RESIGNAL and the BEGIN statement:

```
CREATE PROCEDURE DBA.error_reporting_procedure()
BEGIN
        SELECT *
        FROM sa_error_stack_trace();
END;

CREATE PROCEDURE DBA.proc1()
BEGIN
    BEGIN
        DECLARE v INTEGER = 0;
        SET v = 1 / v;
    EXCEPTION WHEN OTHERS THEN
        CALL DBA.proc2();
    END
END
EXCEPTION WHEN OTHERS THEN
    CALL DBA.error_reporting_procedure();
END;

CREATE PROCEDURE DBA.proc2()
BEGIN
    CALL DBA.proc3();
END;

CREATE PROCEDURE DBA.proc3()
BEGIN
    RESIGNAL;
END;
```

When the procedure above is invoked using CALL proc1(), the following result set is produced:

StackLevel	UserName	ProcName	LineNumber	IsResignal
1	DBA	proc1	8	0
2	DBA	proc2	3	0
3	DBA	proc3	3	1
4	DBA	proc1	5	0

Transactions and savepoints in procedures

SQL statements in a procedure are part of the current transaction. You can call several procedures within one transaction or have several transactions in one procedure.

COMMIT and ROLLBACK are not allowed within any atomic statement.

Savepoints can be used within a procedure, but a ROLLBACK TO SAVEPOINT statement can never refer to a savepoint before the atomic operation started. Also, all savepoints within an atomic operation are released when the atomic operation completes.

Hiding the contents of a procedure, function, trigger, event, or view

To distribute an application and a database without disclosing the logic contained within procedures, functions, triggers, events, and views, you can obscure the contents of these objects using the SET HIDDEN clause of the ALTER PROCEDURE, ALTER FUNCTION, ALTER TRIGGER, ALTER EVENT and ALTER VIEW statements.

Prerequisites

You must be the owner of the object, have the ALTER ANY OBJECT system privilege, or have one of the following privileges:

Procedures and functions – ALTER ANY PROCEDURE system privilege

Views – ALTER ANY VIEW system privilege

Events – MANAGE ANY EVENT system privilege

Triggers –

ALTER ANY TRIGGER system privilege
ALTER privilege on the underlying table and the CREATE ANY OBJECT system privilege
For triggers on views, you must have the ALTER ANY TRIGGER and ALTER ANY VIEW system privileges

Task

The SET HIDDEN clause obfuscates the contents of the associated objects and makes them unreadable, while still allowing the objects to be used. You can also unload and reload the objects into another database.

The modification is irreversible, and deletes the original text of the object. Preserving the original source for the object outside the database is required.

Note: Setting the preserve_source_format database option to On causes the database server to save the formated source from CREATE and ALTER statements on procedures, views, triggers, and events, and put it in the appropriate system view's source column. In this case both the object definition and the source definition are hidden.

However, setting the preserve_source_format database option to On does *not* prevent the SET HIDDEN clause from deleting the original source definition of the object.

Use the appropriate ALTER statement with the SET HIDDEN clause.

Option	Action
Hide an individual object	Execute the appropriate ALTER statement with the SET HIDDEN clause to hide a single procedure, function, trigger, event, or view.
Hide all objects of a specific type	Execute the appropriate ALTER statement with the SET HIDDEN clause in a loop to hide all procedures, functions, triggers, events, or views.

An automatic commit is executed. The object definition is no longer visible. The object can still be directly referenced, and is still eligible for use during query processing.

Statements allowed in procedures, triggers, events, and batches

Most SQL statements are acceptable in batches, with the exception of the following:

ALTER DATABASE (syntax 3 and 4)
CONNECT
CREATE DATABASE
CREATE DECRYPTED FILE
CREATE ENCRYPTED FILE
DISCONNECT
DROP CONNECTION
DROP DATABASE
FORWARD TO
Interactive SQL statements such as INPUT or OUTPUT
PREPARE TO COMMIT
STOP SERVER

You can use COMMIT, ROLLBACK, and SAVEPOINT statements within procedures, triggers, events, and batches with certain restrictions.

SELECT statements used in batches

You can include one or more SELECT statements in a batch. For example:

```
IF EXISTS( SELECT *
           FROM SYSTAB
           WHERE table_name='Employees' )
THEN
   SELECT   Surname AS LastName,
            GivenName AS FirstName
   FROM Employees;
   SELECT Surname, GivenName
   FROM Customers;
   SELECT Surname, GivenName
   FROM Contacts;
END IF;
```

The alias for the result set is necessary only in the first SELECT statement, as the server uses the first SELECT statement in the batch to describe the result set.

A RESUME statement is necessary following each query to retrieve the next result set.

EXECUTE IMMEDIATE used in procedures, triggers, user-defined functions, and batches

The EXECUTE IMMEDIATE statement allows statements to be constructed using a combination of literal strings (in quotes) and variables. For example, the following procedure includes an EXECUTE IMMEDIATE statement that creates a table.

```
CREATE PROCEDURE CreateTableProcedure(
      IN tablename CHAR(128) )
BEGIN
   EXECUTE IMMEDIATE 'CREATE TABLE '
   || tablename
   || '( column1 INT PRIMARY KEY )'
END;
```

The EXECUTE IMMEDIATE statement can be used with queries that return result sets. You use the WITH RESULT SET ON clause with the EXECUTE IMMEDIATE statement to indicate that the statement returns a result set—the default behavior is that the statement does not return a result set. Specifying WITH RESULT SET ON or WITH RESULT SET OFF affects both what happens when the procedure is created, as well as what happens when the procedure is executed.

Consider the following procedure:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

While the procedure definition does not include a RESULT SET clause, the database server tries to determine if the procedure generates one. Here, the EXECUTE IMMEDIATE

statement specifies that a result set is not generated. Consequently, the database server defines the procedure with no result set columns, and no rows exist in the SYSPROCPARM system view for this procedure. A DESCRIBE on a CALL to this procedure would return no result columns. If an embedded SQL application used that information to decide whether to open a cursor or execute the statement, it would execute the statement and then return an error.

As a second example, consider a modified version of the above procedure:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1';
END;
```

Here, the WITH RESULT SET ON clause causes a row to exist for this procedure in the SYSPROCPARM system view. The database server does not know what the result set looks like—because the procedure is using EXECUTE IMMEDIATE—but it knows that one is expected, so the database server defines a dummy result set column in SYSPROCPARM to indicate this, with a name of "expression" and a type of SMALLINT. Only *one* dummy result set column is created; the server cannot determine the number and type of each result set column when an EXECUTE IMMEDIATE statement is being used. Consequently, consider this slightly modified example:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET ON 'SELECT 1, 2, 3';
END;
```

Here, while the SELECT returns a result set of three columns, the server still only places one row in the SYSPROCPARM system view. Hence, this query

```
SELECT * FROM test_result_clause();
```

fails with SQLCODE -866, as the result set characteristics at run time do not match the placeholder result in SYSPROCPARM.

To execute the query above, you can explicitly specify the names and types of the result set columns as follows:

```
SELECT * FROM test_result_clause() WITH (x INTEGER, y INTEGER, z
INTEGER);
```

At execution time, if WITH RESULT SET ON is specified, the database server handles an EXECUTE IMMEDIATE statement that returns a result set. However, if WITH RESULT SET OFF is specified or the clause is omitted, the database server *still* looks at the type of the first statement in the parsed string argument. If that statement is a SELECT statement, it returns a result set. Hence, in the second example above:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF 'SELECT 1';
END;
```

this procedure can be called successfully from Interactive SQL. However, if you change the procedure so that it contains a batch, rather than a single SELECT statement:

```
CREATE OR REPLACE PROCEDURE test_result_clause()
BEGIN
    EXECUTE IMMEDIATE WITH RESULT SET OFF
    'begin declare v int; set v=1; select v; end';
END;
```

then a CALL of the test_result_clause procedure returns an error (SQLCODE -946, SQLSTATE 09W03).

This last example illustrates how you can construct a SELECT statement as an argument of an EXECUTE IMMEDIATE statement within a procedure, and have that procedure return a result set.

```
CREATE PROCEDURE DynamicResult(
   IN Columns LONG VARCHAR,
   IN TableName CHAR(128),
   IN Restriction LONG VARCHAR DEFAULT NULL )
BEGIN
    DECLARE Command LONG VARCHAR;
    SET Command = 'SELECT ' || Columns || ' FROM ' || TableName;
    IF ISNULL( Restriction,'') <> '' THEN
         SET Command = Command || ' WHERE ' || Restriction;
    END IF;
    EXECUTE IMMEDIATE WITH RESULT SET ON Command;
END;
```

If the procedure above is called as follows:

```
CALL DynamicResult(
   'table_id,table_name',
   'SYSTAB',
   'table_id <= 10');
```

it yields the following result:

table_id	table_name
1	ISYSTAB
2	ISYSTABCOL
3	ISYSIDX
...	...

The CALL above correctly returns a result set, even though the procedure utilizes EXECUTE IMMEDIATE. Some server APIs, such as ODBC, utilize a PREPARE-DESCRIBE-EXECUTE-OR-OPEN combined request that either executes or opens the statement, depending on if it returns a result set. Should the statement be opened, the API or application can subsequently issue a DESCRIBE CURSOR to determine what the actual result set looks like, rather than rely on the content of the SYSPROCPARM system view from when the procedure was created. Both DBISQL and DBISQLC use this technique. In these cases, a CALL of the procedure above executes without an error. However, application interfaces that rely on the statement's DESCRIBE results will be unable to handle an arbitrary statement.

In ATOMIC compound statements, you cannot use an EXECUTE IMMEDIATE statement that causes a COMMIT, as COMMITs are not allowed in that context.

Automate Tasks Using Schedules and Events

Use scheduling and event handling features to automate database administration and other tasks.

Task automation using schedules and events

Many database administration tasks are best performed systematically. For example, a regular backup procedure is an important part of proper database administration procedures.

You can automate routine tasks in SAP Sybase IQ by adding an **event** to a database, and providing a schedule for the event. Whenever one of the times in the schedule passes, the database server runs a sequence of actions called an **event handler**.

Database administration also requires taking action when certain conditions occur. For example, it may be appropriate to email a notification to a system administrator when a disk containing the transaction log is filling up so that the administrator can handle the situation. These tasks too can be automated by defining event handlers for one of a set of **system events**.

Events

You can automate routine tasks in SAP Sybase IQ by adding an event to a database, and providing a schedule for the event. SAP Sybase IQ supports the following types of events:

- **Scheduled events –** have an associated schedule and execute at specified times.
- **System events –** are associated with a particular type of condition that is tracked by the database server.
- **Manual events –** are fired explicitly using the TRIGGER EVENT statement.
- **User trace events –** are used to log information about an application to an event tracing session. These events are visible to all connections to a database.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

Schedules

By scheduling activities you can ensure that a set of actions is executed at a set of preset times. The scheduling information and the event handler are both stored in the database itself.

Although this is not usually necessary, you can define complex schedules by associating more than one schedule with a named event. For example, a retail outlet might want an event to occur once per hour during hours of operation, where the hours of operation vary based on the day of the week. You can achieve the same effect by defining multiple events, each with its own schedule, and by calling a common stored procedure.

When scheduling events, you can use either full-length English day names (Monday, Tuesday, and so on) or the abbreviated forms of the day (Mon, Tue, and so on). You must use the full-length English day names if you want the day names to be recognized by a server running in a language other than English.

The following examples give some ideas for scheduled actions that may be useful.

Examples

Summarize orders at the end of each business day. This example uses iqdemo.db.

```
CREATE TABLE OrderSummary(c1 date, c2 int);
CREATE EVENT SummarizeSchedule
START TIME '6:00 pm'on ('Mon', 'Tue', 'Wed', 'Thu', 'Fri')
HANDLER
  BEGIN
    INSERT INTO DBA.OrderSummary
      SELECT MAX(OrderDate),
      COUNT(*)
  FROM GROUPO.SalesOrders
  WHERE OrderDate = current date
END;
```

Schedule definition

To permit flexibility, schedule definitions have several components to them:

- **Name –** Each schedule definition has a name. You can assign more than one schedule to a particular event, which can be useful in designing complex schedules.
- **Start time –** You can define a start time for the event, which is the time when execution begins.
- **Range –** As an alternative to a start time, you can specify a range of times for which the event is active. The event occurs between the start and end time specified. Frequency is determined by the specified recurrence.
- **Recurrence –** Each schedule can recur. The event is triggered on a frequency that can be given in hours, minutes, or seconds on a set of days that can be specified as days of the week or days of the month. Recurring events include an `EVERY` or `ON` clause.

System events

SAP Sybase IQ tracks several system events. Each system event provides a hook on which you can hang a set of actions. The database server tracks the events for you, and executes the actions (as defined in the event handler) when the system event satisfies a provided **trigger condition**.

By defining event handlers to execute when a chosen system event type occurs and satisfies a trigger condition that you define, you can improve the security and safety of your data, and help ease administration. The actions of an event handler are committed if no error is detected during execution, and rolled back if errors are detected.

System event types

- **BackupEnd –** You can use the BackupEnd event type to take action at the end of a backup.
- **Connection events –** When a connection is made (Connect) or when a connection attempt fails (ConnectFailed). You may want to use these events for security purposes. As an alternative to a connect event handler, you may want to consider using a login procedure.
- **DatabaseStart –** You can use the DatabaseStart event type to take action when a database is started.
- **Deadlock –** You can use the Deadlock event to take action when a deadlock occurs. The event handler can use the sa_report_deadlocks procedure to obtain information about the conditions that led to the deadlock. When using the Deadlock event, you should configure the database server to capture deadlock information by setting the log_deadlocks option to On, and by enabling the RememberLastStatement feature using sa_server_option or the -zl server option.

  Deadlock events fire for connection deadlocks and thread deadlocks. A deadlock event provides no information beyond what is available via the sa_report_deadlocks system procedure. However, using this event allows you to act on the deadlock in a timely manner. A quick response may be important since the amount of deadlock-related information the database server maintains is limited.
- **Disconnect –** You can use the Disconnect event to take action when a user or application disconnects.
- **Free disk space –** Tracks the available disk space on the device holding the database file (DBDiskSpace), the log file (LogDiskSpace), or temporary file (TempDiskSpace). This system event is not available on Windows Mobile.

  You may want to use disk space events to alert administrators of a disk space shortage.

  You can specify the -fc option when starting the database server to implement a callback function when the database server encounters a file system full condition.
- **File size –** The file reaches a specified size. This can be used for the database file (GrowDB), the transaction log (GrowLog), or the temporary file (GrowTemp).

  You may want to use file size events to track unusual actions on the database, or monitor bulk operations.
- **GlobalAutoincrement –** When the number of remaining values for a column defined with GLOBAL AUTOINCREMENT is less than one percent of its range, the GlobalAutoincrement event fires. This can be used to request a new value for the global_database_id option based on the table and number of remaining values that are supplied as parameters to this event. To get the remaining values for the table within the event, use the EVENT_PARAMETER function with the RemainingValues and TableName parameters. RemainingValues returns the number of remaining values that can be generated for the column, while TableName returns the table containing the GLOBAL AUTOINCREMENT column that is near the end of its range.

- **RAISERROR error –** When a RAISERROR statement is executed, you can use the RAISERROR event type to take actions. The error number used in the RAISERROR statement can be determined within the event handler using the EVENT_CONDITION function (for example, `EVENT_CONDITION( 'ErrorNumber' )`).
- **Idle time –** The database server has been idle for a specified time (ServerIdle). You may want to use this event type to perform routine maintenance operations at quiet times.

Trigger conditions for events

Each event definition has a system event associated with it. It also has one or more trigger conditions. The event handler is triggered when the trigger conditions for the system event are satisfied.

The trigger conditions are included in the WHERE clause of the CREATE EVENT statement, and can be combined using the AND keyword. Each trigger condition is of the following form:

```
event_condition( condition-name ) comparison-operator value
```

The *condition-name* argument is one of a set of preset strings, which are appropriate for different event types. For example, you can use `DBSize` (the database file size in megabytes) to build a trigger condition suitable for the `GrowDB` system event. The database server does not check that the condition-name matches the event type: it is your responsibility to ensure that the condition is meaningful in the context of the event type.

Examples

- Notify an administrator of a possible attempt to break into the database. This example uses iqdemo.db:

```
create event SecurityCheck
type ConnectFailed
handler
begin
declare num_failures int;
declare mins int;
insert into FailedConnections( log_time )
values ( current timestamp );
select count( * ) into num_failures
from FailedConnections
where log_time >= dateadd(minute, -5, current timestamp );
if( num_failures >= 3 ) then
  select datediff( minute, last_notification, current timestamp )
into mins from Notification;
  if( mins > 30 ) then
    update Notification
    set last_notification = current timestamp;
end if
end if
end;
```

Event handlers

Event handlers execute on a separate connection from the action that triggered the event, and so do not interact with client applications. They execute with the privileges of the creator of the event.

Event handlers, whether for scheduled events or for system event handling, contain compound statements, and are similar in many ways to stored procedures. You can add loops, conditional execution, and so on.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

Context information for event handlers

Unlike stored procedures, event handlers do not take any arguments. You can use the EVENT_PARAMETER function to access information about the context in which an event was triggered. The information returned includes the connection ID and user ID that caused an event to be triggered, and the event name and the number of times it has been executed.

Test event handlers

During development, you want event handlers to be triggered at convenient times. You can use the TRIGGER EVENT statement to explicitly cause an event to execute, even when the trigger condition or scheduled time has not occurred. However, TRIGGER EVENT does not cause disabled event handlers to be executed.

While it is not good practice to develop event handlers on a production database, you can disable event handlers explicitly using the ALTER EVENT statement.

Code sharing

It can be useful to use a single set of actions to handle multiple events. For example, you may want to take a notification action if disk space is limited on any of the devices holding the database or log files. To do this, create a stored procedure and call it in the body of each event handler, passing any needed context information as parameters to the procedure.

Debug event handlers

Debugging event handlers is very similar to debugging stored procedures. The event handlers appear in the events list.

Hide event handlers

You can use the ALTER EVENT statement with the SET HIDDEN clause to hide the definition of an event handler. Specifying the SET HIDDEN clause results in the permanent obfuscation of the event handler definition stored in the action column of the ISYSEVENT system table.

Limit active events
You can also determine how many instances of a particular event handler are currently active using EVENT_PARAMETER function with the NumActive context name. This function is useful to limit an event handler so that only one instance executes at any given time.

How the database server checks for system events

System events are classified according to their **event type**, as specified in the CREATE EVENT statement. There are two kinds of event types:

- **Active event types –** Some event types are the result of action by the database server itself. These active event types include growing database files, or the start and end of different database actions (BackupEnd and so on) or RAISERROR.

  When the database server takes the action, it checks to see whether the trigger conditions defined in the WHERE clause are satisfied, and if so, triggers any events defined for that event type.
- **Polled event types –** Some event types, such as free disk space types (DBDiskSpace and so on) and IdleTime type, are not triggered solely by database actions.

  For these types of events, the database server polls every thirty seconds, starting approximately thirty seconds after the database server is started.

  For the IdleTime event type, the database server checks whether the server has been idle for the entire thirty seconds. If no requests have started and none are currently active, it adds the idle check interval time in seconds to the idle time total; otherwise, the idle time total is reset to 0. The value for IdleTime is therefore always a multiple of thirty seconds. When IdleTime is greater than the interval specified in the trigger condition, event handlers associated with IdleTime are fired.

How the database server checks for scheduled events

The calculation of scheduled event times is done when the database server starts, and each time a scheduled event handler completes.

The calculation of the next scheduled time is based on the increment specified in the schedule definition, with the increment being added to the previous start time. If the event handler takes longer to execute than the specified increment, so that the next time is earlier than the current time, the database server increments until the next scheduled time is in the future.

For example, an event handler that takes sixty-five minutes to execute and is requested to run every hour between 9:00 and 5:00 will run every two hours, at 9:00, 11:00, 1:00, and so on.

To run a process such that it operates between 9:00 and 5:00 and delays for some period before the next execution, you could define a handler to loop until its completion time has passed, with a WAITFOR statement between each iteration.

If you are running a database server intermittently, and it is not running at a scheduled time, the event handler does not run at startup. Instead, the next scheduled time is computed at startup.

If, for example, you schedule a backup to take place every night at one o'clock, but regularly shut down the database server at the end of each work day, the backup never takes place.

If the next scheduled execution of an event is more than one hour away, the database server will recalculate its next scheduled time on an hourly basis. This allows events to fire when expected when the system clock is adjusted because of a change to or from Daylight Savings Time.

How event handlers are executed

When an event handler is triggered, a temporary internal connection is made on which the event handler is executed. The handler is not executed on the connection that caused the handler to be triggered, so statements such as MESSAGE...TO CLIENT, which interact with the client application, are not meaningful within event handlers. Similarly, statements that return result sets are not permitted.

The temporary connection on which the handler is executed does not count towards the connection limit for licensing purposes, and the procedure specified by the login_procedure option is not executed for event connections.

Event handlers execute on a separate connection, with the privileges of the event owner. You can also call a procedure from within the event handler, in which case the procedure executes with the privileges of procedure owner. The separate connection for the event handler does not count towards the ten-connection limit of the personal database server.

Any event errors are logged to the database server message log.

Note: The transaction in an event handler is committed if no errors are detected during execution, and rolled back if errors are detected.

Hiding an event handler

For improved security, you can hide the definition for an event handler using the ALTER EVENT statement.

Prerequisites

You must have either the MANAGE ANY EVENT or ALTER ANY OBJECT system privilege.

Task

1. Connect to the database.
2. Execute an ALTER EVENT statement with the SET HIDDEN clause:

```
ALTER EVENT event-name SET HIDDEN
```

The event handler is permanently obfuscated in the event handler definition that is stored in the action column of the ISYSEVENT system table.

Event handlers

Event handlers execute on a separate connection from the action that triggered the event, and so do not interact with client applications. They execute with the privileges of the creator of the event.

Event handlers, whether for scheduled events or for system event handling, contain compound statements, and are similar in many ways to stored procedures. You can add loops, conditional execution, and so on.

After each execution of an event handler, a COMMIT occurs if no errors occurred. A ROLLBACK occurs if there was an error.

Context information for event handlers

Unlike stored procedures, event handlers do not take any arguments. You can use the EVENT_PARAMETER function to access information about the context in which an event was triggered. The information returned includes the connection ID and user ID that caused an event to be triggered, and the event name and the number of times it has been executed.

Test event handlers

During development, you want event handlers to be triggered at convenient times. You can use the TRIGGER EVENT statement to explicitly cause an event to execute, even when the trigger condition or scheduled time has not occurred. However, TRIGGER EVENT does not cause disabled event handlers to be executed.

While it is not good practice to develop event handlers on a production database, you can disable event handlers explicitly using the ALTER EVENT statement.

Code sharing

It can be useful to use a single set of actions to handle multiple events. For example, you may want to take a notification action if disk space is limited on any of the devices holding the database or log files. To do this, create a stored procedure and call it in the body of each event handler, passing any needed context information as parameters to the procedure.

Debug event handlers

Debugging event handlers is very similar to debugging stored procedures. The event handlers appear in the events list.

Hide event handlers

You can use the ALTER EVENT statement with the SET HIDDEN clause to hide the definition of an event handler. Specifying the SET HIDDEN clause results in the permanent obfuscation of the event handler definition stored in the action column of the ISYSEVENT system table.

Limit active events
You can also determine how many instances of a particular event handler are currently active using EVENT_PARAMETER function with the NumActive context name. This function is useful to limit an event handler so that only one instance executes at any given time.

Retrieving Information About an Event or Schedule

SAP Sybase IQ stores information about events, system events, and schedules in the system tables SYSEVENT, SYSEVENTTYPE, and SYSSCHEDULE.

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

You can list event names by querying the SYSEVENTsytem table:

```
SELECT event_id, event_name FROM SYSEVENT
```

You can list schedule names by querying the SYSSCHEDULE system table:

```
SELECT event_id, sched_name FROM 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.

Audit Database Events

Auditing records database events in the transaction log.

dbtran Database Administration Utility

Use the **dbtran** log translation utility, at the command prompt, to translate a transaction log into a `.sql` command file.

Syntax

Running against a database server:

```
dbtran [ options ] -c { connection-string } -n SQL-file
```

Running against a transaction log:

```
dbtran [ options ] [ transaction-log ] [ SQL-file ]
```

Parameters

Option	Description
@data	Reads in options from the specified environment variable or configuration file.
-a	Controls whether uncommitted transactions appear in the transaction log. The transaction log contains changes made only before the most recent **COMMIT** by any transaction. If you do not specify **-a**, only committed transactions appear in the output file. If you specify **-a**, any uncommitted transactions found in the transaction log appear.
-c *"keyword=value; ..."*	Specifies the connection string when running the utility against a database server.
-d	Specifies that transactions are written in order from earliest to latest. This feature is intended for auditing database activity: do not apply **dbtran** output against a database.

Option	Description
-ek *key*	Specifies the encryption key for strongly encrypted databases. If you have a strongly encrypted database, you must provide the encryption key to use the database or transaction log. Specify either **-ek** or **-ep**, but not both. The command fails if you do not specify the correct encryption key. If you are running **dbtran** against a database server using the **-c** option, specify the key using a connection parameter instead of using the **-ek** option. For example, the following command gets the transaction log information about database **enc.db** from the database server sample, and saves its output in `log.sql`. **dbtran -n log.sql -c "ENG=sample;DBF=enc.db;UID=DBA;PWD=sql;DBKEY=mykey"**
-ep	Prompts for the encryption key. This option causes a window to appear, in which you enter the encryption key. It provides an extra measure of security by never allowing the encryption key to be seen in clear text. Specify either **-ek** or **-ep**, but not both. The command fails if you do not specify the correct encryption key. If you are running **dbtran** against a database server using the **-c** option, specify the key using a connection parameter, instead of using the **-ep** option. For example, the following command gets the transaction log information about database **enc.db** from the database server sample, and saves its output in `log.sql`. **dbtran -n log.sql -c "ENG=sample;DBF=enc.db;UID=DBA;PWD=sql;DBKEY=mykey"**
-f	Outputs only transactions completed since the last checkpoint.
-g	Adds auditing information to the transaction log if the auditing database option is turned on.
-ir *offset1,offset2*	Outputs a portion of the transaction log between two specified offsets.

Option	Description
-is *source,...*	Outputs operations on rows that have been modified by operations from one or more of the following sources, specified as a comma-separated list: • All — all rows. This is the default setting. • SQLRemote — include only rows that were modified using SQL Remote. You can also use the short form “SR”. • RepServer — include only rows that were modified using the Replication Agent (LTM) and Replication Server. You can also use the short form “RS”. • Local — include only rows that are not replicated.
-it *owner.table,...*	Outputs operations on the specified, comma-separated list of tables. Specify each table as owner.table.
-j date/time	Translates only transactions from the most recent checkpoint prior to the given date or time. The user-provided argument can be a date, time, or date and time, enclosed in quotes. you omit a time, the default is 00:00. If you omit a date, the current day is the default. The acceptable format for the date and time is: "YYYY/MMM/DD HH:NN".
-k	Prevents partial `.sql` files from being erased if an error is detected. If an error is detected while **dbtran** is running, the `.sql` file generated until that point is normally erased to ensure that a partial file is not used. Specifying this option may be useful if you are attempting to salvage transactions from a damaged transaction log.
-m	Specifies a directory that contains transaction logs. Use this option with the **-n** option.
-n *filename*	Specifies the output file that holds the SQL statements when you run **dbtran** against a database server.
-o *filename*	Writes output messages to the named file.
-r	Removes any uncommitted transactions. This is the default behavior.
-rsu *username,...*	Specifies a comma-separated list of user names to override the default Replication Server user names. By default, the **-is** option assumes the default Replication Server user names of dbmaint and sa.

Option	Description
-s	Controls how **UPDATE** statements are generated. If you do not use this option, and there is no primary key or unique index on a table, **dbtran** generates **UPDATE** statements with a nonstandard FIRST keyword in case of duplicate rows. If you do use this option, the FIRST keyword is omitted for compatibility with the SQL standard.
-sr	Includes in the output file generated comments describing how SQL Remote distributes operations to remote sites.
-t	Controls whether triggers are included in the command file. By default, actions performed by triggers are not included in the command file. If the matching trigger is in the database, when the command file is run against the database, the trigger performs the actions automatically. Trigger actions should be included if the matching trigger does not exist in the database against which the command file is to run.
-u *userid,...*	Limits the output from the transaction log to include only specified users.
-x *userid,...*	Limits the output from the transaction log to exclude specified users.
-y	Replaces existing command files without prompting for confirmation. If you specify **-q**, you must also specify **-y** or the operation fails.
transaction-log	Specifies the log file to be translated. Cannot be used with **-c** or **-m** options.
SQL-file	Names the output file containing the translated information. For use with *transaction-log* only.

Usage

You can run **dbtran**:

- Against a database server — connects to the database server using the connection string specified following the **-c** option, and places output in a file specified with the **-n** option. The BACKUP DATABASE system privilege is required to run in this way. For example, this command translates log information from the **iqdemo** server and places the output in a file named `iqdemo.sql`:

```
dbtran -c "eng=iqdemo;dbn=iqdemo;dbf=iqdemo.db;uid=DBA;pwd=sql" -n
iqdemo.sql
```

- Against a transaction log file — acts directly against a transaction log file. Protect your transaction log file from general access to prevent users from running this statement.

```
dbtran iqdemo.log iqdemo.sql
```

dbtran shows the earliest log offset in the transaction log, which you can use to determine the order in which multiple log files were generated.

dbtran-c attempts to translate the online transaction log file, and all the offline transaction log files in the same directory as the online transaction log file. If the directory contains transaction log files for more than one database, you may see an error. To avoid this, ensure that each directory contains transaction log files for only one database.

A transaction can span multiple transaction logs. If transaction log files contain transactions that span logs, translating a single transaction log file (for example, `dbtran demo.log`) might lose the spanning transactions. For **dbtran** to generate complete transactions, use the **-c** or **-m** options with the transaction log files in the directory.

Exit codes are 0 (success) or nonzero (failure).

This utility accepts @filename parameters.

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.

Troubleshooting Hints

SAP Sybase IQ provides many resources for addressing problems.

Sources of Online Support

If you cannot resolve a problem using documentation, see the SAP Sybase IQ online support Web site, MySybase.

MySybase lets you search closed support cases, software bulletins, and resolved and known problems, using a view customized for your needs. You can even open a Technical Support case online.

You can use MySybase from most Internet browsers. Point your Web browser to *MySybase* for information on how to sign up for and use this free service. For additional useful SAP Sybase Web sites, see the *Release Bulletin*.

Solutions for Specific Conditions

More information may be needed to diagnose and resolve certain issues. You can use diagnostic tools to diagnose various conditions

Decision Flow for Server Recovery and Database Repair

You may experience trouble starting a server or database, or connecting to or verifying a database.

1. Does the server start?

   If yes, go to step 2.

   If no, see *Server Operational Issues*. If you cannot start the server after following these suggestions in this section, see *Starting a Server in Forced Recovery Mode* and start the server in forced recovery mode.

   If the server does not start in forced recovery mode, call Technical Support. You may need to restore the database from backup.
2. Can you connect to the database?

   If you cannot connect to the database, see *Database Connection Issues* for troubleshooting suggestions.

   If you can connect to the database and you previously started the server in forced recovery, see *Analysis of Allocation Problems* for information on verifying database allocation and recovering leaked blocks.

If you can connect to the database, but suspect the database may be inconsistent, see *Database Verification* for information on checking the consistency of your database.

3. The server is running and you can connect, but you want to verify the consistency of your database.

   If you previously started the server with forced recovery or you suspect database inconsistency, run DBCC checks to validate the database. See *Database Verification* for information on checking both index consistency and database allocation.

4. The server is running, you can connect, you have run DBCC checks, and you need to repair the index inconsistencies or allocation problems detected by DBCC.

   If **sp_iqcheckdb** reports errors in the Index Summary and Index Statistics sections of the results, see *Index Error Repair* for the procedure to repair index problems using DBCC.

   If **sp_iqcheckdb** reports errors in the Allocation Summary and Allocation Statistics sections of the results, see *Repairing Allocation Problems using DBCC* for the procedure to repair allocation problems using DBCC.

Server Operational Issues

Issues that may affect server operation include startup, shutdown, unresponsiveness, and abnormal termination.

SAP Sybase IQ Will Not Start

If there is a problem starting the server, **start_iq** returns a non zero value.

If you did not specify a log file after the **-o** switch on startup, SAP Sybase IQ writes the error to the first one of the following that is defined:

- `$IQDIR16/logfiles/<servername>.nnnn.stderr`
- `$IQDIR16/logfiles/<servername>.nnnn.srvlog`
- The Systems applications log file

There are several possible causes.

Transaction Log File Does Not Match the Database

Messages appear in the server log file (`.srvlog`) and in the window where you are starting the server:

```
Starting database "dbname" (/dbdir/dbname.db)
at Fri Apr 27 2009 10:53 Transaction log: dbname.log
Error: Cannot open transaction log file
-- Can't use log file "dbname.log" since the database
file has been used more recently
Cannot open transaction log file
-- Can't use log file "dbname.log" since the database
file has been used more recently
Database server stopped at Fri Apr 27 2009 10:53
```

If these errors are reported when you are starting the server, verify that the server is using the correct transaction log file. If you cannot find the correct transaction log file, the safest way to recover from this situation is to restore from the last valid backup.

If you cannot find the correct transaction log and restoring from backup is not an option, perform an emergency recovery without a transaction log.

Server Cannot Find the Transaction Log

If the server fails to start because it cannot find the transaction log, messages appear in the server log file.

```
Transaction log: /dbdir/dbname.log...
Error: Cannot open transaction log file
-- No such file or directory
Cannot open transaction log file
-- No such file or directory
```

If this error is reported when you attempt to start the server, find the transaction log file and copy the file to the same directory as the database `.db` file. If you cannot find the correct transaction log file, restore from the last valid backup.

If no other option for starting the server is available, you may be able to start the server using the emergency recovery **-f** option. Contact SAP Sybase Technical Support for assistance, if necessary.

Warning! This procedure is highly risky and is not recommended except in extreme cases.

Server Name Is Not Unique on Your Network

If multiple servers on your system have the same name, messages appear in the server log file (`*.srvlog` or the name specified in the **-o** startup option) when you attempt to start the server using **start_iq**.

```
DBSPAWN ERROR:  -85
Communication error
```

If you see these errors in the server log file and the server does not start, try to start the server using the **iqsrv16** command. The **iqsrv16** command returns a more specific error message:

```
A database server with that name has already started
```

Once you have verified that the problem is a duplicate server name on your network, start the server with a name that is different from the names of servers that are already running.

Log File Has Illegal Name

If you specified a separate request-level logging file, but the file name is an illegal identifier, errors result on server startup.

```
Naming conflict: "iqdemo" --
aborting
```

```
Database naming conflict --
aborting startup
```

These errors may indicate a space in the file path specified on the **-zo** option.

Specify the **-zo** option again and enclose any file name that contains a space within quotation marks.

Server Port Number Is Not Unique on the Machine

If an SAP Sybase IQ server is running and you attempt to start another SAP Sybase IQ server on the same machine using the same port number, messages appear in the server log file (`*.srvlog`).

```
Trying to start TCPIP link ...
TCPIP communication link not started
Unable to initialize requested communication links
...
DBSPAWN ERROR:  -85
Communication error

Server failed to start
```

If you see these messages in the server log file and the server does not start, run the **stop_iq** command (UNIX) to display the names and port numbers of SAP Sybase IQ servers already running on the machine. Then try to start your server, specifying either a port number that is not in use or no port number. When you start a server and do not provide a port number (and the default port number is already in use), SAP Sybase IQ generates an available port number.

You see these messages in the server log file when you start the server without specifying a port number:

```
Trying to start TCPIP link ...
Unable to start on default port; starting on port
49152 instead
TCPIP link started successfully
Now accepting requests
...
Server started successfully
```

Server Started with an Incorrect Path

When you start a new multiplex server, the database file path must match the database file path specified when creating that server.

If you use the wrong path, server startup fails, and writes these messages in the server log file (`*.srvlog`):

```
E. 08/18 07:22:19. MPX: server myserver
has been started with an incorrect catalog path
(expected path: /work/IQ-16_0/demo/mympx/iqdemo.db).
-- (st_database.cxx 7883)
I. 08/18 07:22:19. Database server shutdown due
to startup error
```

```
DBSPAWN ERROR:  -82
Unable to start specified database: autostarting
database failed
```

If you see these messages, restart the server with the expected path. If you plan to use UNIX soft (symbolic) links for server paths, you must create the soft link before you run **CREATE MULTIPLEX SERVER**.

Environment Variables Not Set Correctly

If your database configuration file parameters differ from those used by **start_iq**, make sure the correct parameters are used to start the server.

You Cannot Run start_iq

If you cannot run the **start_iq** command and you normally use a configuration file or other command line switches, try starting the server using only **start_iq** with the server name and database name.

If the server starts with this simple command, then the problem is probably caused by one or more of the switches or parameters entered on the command line or in the configuration file. Try to isolate which parameter or switch is preventing the server from starting.

If the server does not start with the most basic **start_iq** command, try starting the `iqdemo` demo database using your configuration file and command line switches. If the server starts with the `iqdemo` database, there may be a problem with your database.

If you still cannot run the **start_iq** command, use the **iqsrv16** command.

Note: Use **iqsrv16** only for troubleshooting server start up errors. Always use **start_iq** to start SAP Sybase IQ servers.

Before running **iqsrv16**, you must perform the following tasks (which **start_iq** normally does for you):

- Remove all limits, and then set limits on the stack size and descriptors. To do so, go to the C shell and issue these commands:

  ```
  % unlimit
  % limit stacksize 8192
  % limit descriptors 4096
  ```

  Note: unlimit affects soft limits only. You must change any hard limits by setting kernel parameters.
- Set all server options appropriately for your platform. See the *Installation and Configuration Guide* guide.
- Add the path `$SYBASE/OCS-15_0/lib` to the environment to load the engine and required libraries before you invoke **iqsrv16**. Put this path in the environment only during testing, as follows:

  On AIX:

  ```
  % setenv LIBPATH "${LIBPATH}:{SYBASE}/OCS-15_0/lib"
  ```

On other UNIX/LINUX platforms:

```
% setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${SYBASE}/OCS-15_0/
lib"
```

For any database created with a relative path name, you must start the database server from the directory where the database is located.

Note what directory you are in when you start the server. The server startup directory determines the location of any new database files you create with relative path names. If you start the server in a different directory, SAP Sybase IQ cannot find those database files.

Any server startup scripts should change to a known location before issuing the server startup command.

The syntax for **iqsrv16** is:

```
iqsrv16 -n server-name -gm number
[ other-server-switches ] [ database-file [ database-switches ] ]
```

Note: On the **iqsrv16** command line, the last option specified takes precedence, so to override your configuration file, list any options you want to change after the configuration file name. For example:

```
iqsrv16 @iqdemo.cfg -x 'tcpip{port=1870}' iqdemo
```

The **–x** parameter here overrides connection information in the `iqdemo.cfg` file.

If the server fails to start when you run the **iqsrv16** command, then attempt to start again using the **iqsrv16** utility with minimal switches and parameters. For example:

```
iqsrv16 -n <servername> <dbname>.db -c 32m
-gd all -gl all
```

If the server starts with the minimum parameters and switches, then one of the parameters or switches normally used to start the server may be causing a problem. Try to isolate which parameter or switch is preventing the server from starting.

When you start the server with the **iqsrv16** command, it does not run in the background, and messages do not automatically go to the server log. However, if you include the **-o** file name server switch, messages are sent to the named file in addition to the server window.

SAP Sybase IQ Stops Processing or Stops Responding

You can detect the cause of server unresponsiveness by looking in the SAP Sybase IQ message file.

Possible Causes

The most common causes of server unresponsiveness include:

- Insufficient disk space
- Insufficient room in main or temp buffer cache

Action

If your server seems to be prone to unresponsiveness, either while processing or during shutdown, use the **start_iq** command line option **-z** and the SAP Sybase IQ database option `QUERY_PLAN = 'ON'` to log useful information in the SAP Sybase IQ message (`.iqmsg`) and server log (`.srvlog`) files.

In addition to logging this information, there are other steps you can take to determine the cause of the problem:

- Check both the SAP Sybase IQ message file and the server log file for `You have run out of space...` messages. If you have run out of IQ main store or IQ temporary store, add the appropriate dbspace with the **CREATE DBSPACE** command.
  Setting the database options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DB_SPACE_MB` to large enough values to handle running out of space during a DDL **COMMIT** or **CHECKPOINT** is also important. A few hundred MB should be enough, but you can set these options higher for a large database.
- Determine if the SAP Sybase IQ server process (`iqsrv16`) is consuming CPU cycles by monitoring the CPU usage for a few minutes at the operating system level. Record this information. If the CPU usage changes, then the SAP Sybase IQ server process should be processing normally.
  If the SAP Sybase IQ server CPU usage is normal, you can examine what the server is doing, that is, what statement the server is currently executing.
- If there are no out of space indications, use Interactive SQL on a new or existing connection to gather the following information, in the specified order.

Table 23. Information to Gather for Server Unresponsiveness

Command	Informational purpose
SELECT db_name()	Database name
CHECKPOINT	Checkpoint can succeed
sa_conn_properties ># sa_conn_properties.out	Connection information
sa_conn_info ># sa_conn_info.out	Connection information
sa_db_properties ># sa_db_properties.out	Database property information
sa_eng_properties ># sa_eng_properties.out	Server property information
sp_iqstatus ># sp_iqstatus.out	Database status information
sp_iqconnection ># sp_iqconnection.out	Connection information
sp_iqtransaction ># sp_iqtransaction.out	Transaction information

If you cannot resolve the issue, contact SAP Sybase Technical Support for assistance. They can use the information you have just gathered to help diagnose the problem.

- When the server is unresponsive, generate a stack trace for each SAP Sybase IQ thread by creating a file named `DumpAllThreads` or `dumpallthreads` in the `$IQDIR16/logfiles` directory (the `%ALLUSERSPROFILE\%\SybaseIQ\logfiles` folder on Windows 64 platforms, `C:\ProgramData\SybaseIQ\logfiles` for Vista 64).

  Starting SAP Sybase IQ as recommended, using the Program Manager or **start_iq** command, sets the *IQDIR16* variable automatically. If the *IQDIR16* variable is not set, create the `DumpAllThreads` file in the directory in which `iqsrv16` was started.

  The SAP Sybase IQ server detects the presence of the `DumpAllThreads` file and writes a stack trace for each IQ thread in the stack trace file `stktrc-YYYYMMDD-HHNNSS_#.iq`. After the stack traces are written to the stack trace file, the `DumpAllThreads` file is deleted.

  This stack trace information can be used by SAP Sybase Technical Support to help diagnose the problem.
- If you can connect to the database, run the **IQ UTILITIES** buffer cache monitor on the main and temp (private) buffer caches for ten minutes with a ten-second interval:
  1. Connect to the database or use the existing connection.
  2. `CREATE TABLE #dummy_monitor(c1 INT);`
  3. `IQ UTILITIES MAIN INTO #dummy_monitor START MONITOR '-append -debug -interval 10 -file_suffix iqdbgmon';`
  4. `IQ UTILITIES PRIVATE INTO #dummy_monitor START MONITOR '-append -debug -interval 10 -file_suffix iqdbgmon';`

     Let the process run for 10 minutes, then stop the buffer cache monitor:
  5. `IQ UTILITIES MAIN INTO #dummy_monitor STOP MONITOR;`
  6. `IQ UTILITIES PRIVATE INTO #dummy_monitor STOP MONITOR;`
- Check near the end of the SAP Sybase IQ message file for the message `Resource count 0`, which may be followed by an `Open Cursor` message. These messages indicate a resource depletion, which can cause a deadlock. The immediate solution is to reduce the number of active connections using Ctrl+C or the **DROP CONNECTION** command.

  The long-term solution for avoiding deadlocks due to resource depletion is one or a combination of:
  - Restricting the number of users on the server by reducing the value of the **-gm** server startup option
  - Adding another secondary server to a multiplex
  - Increasing the processing capacity of the hardware by adding CPUs

System Failure/SAP Sybase IQ Failure

You can detect the cause of system/SAP Sybase IQ failure by looking in the SAP Sybase IQ message file.

Possible Causes

Various.

Actions

- Copy or rename the message log file (`dbname.iqmsg`) before trying to restart the database. This ensures that any useful information in the file is not lost.
- On UNIX, send a copy of the stack trace to SAP Sybase Technical Support. The stack trace should be in the directory where you started the database server, in a file named `stktrc-YYYYMMDD-HHNNSS_#.iq`. If the database was open when the failure occurred, the stack trace should also be in the SAP Sybase IQ message log (default name `dbname.iqmsg`). This information helps SAP Sybase Technical Support determine why the failure occurred.
- Restart the server with the **start_iq** command. When the database restarts, recovery occurs automatically.
- Try to start the server without starting a database. If you can start the server but not the database, check that database parameters are specified correctly on the startup line and in the connection profile.
- If you query catalog store tables extensively, restart the server and make sure that the `TEMP_SPACE_LIMIT_CHECK` option is on. With this option setting, if a connection exceeds its quota of catalog store temporary file space, it receives a non fatal error.

Server Fails to Shut Down

To shut down the server, run the **dbstop** utility or **stop_iq**, type `q` in the server window on UNIX, or click **Shutdown** on the server window on Windows.

Possible Causes

Various.

Actions

Perform these actions if the server fails to shut down.

On UNIX systems:

1. Capture **ps** operating system utility output, so you can submit this output to Technical Support. On Sun Solaris, two different **ps** options are available. Use both.

   ```
   ps -aAdeflcj|egrep "PPID|iqsrv16"
   ```

   ```
   /usr/ucb/ps -awwwlx|egrep "PPID|iqsrv16"
   ```

2. Try to kill the process at the operating system level to generate a core dump.

   ```
   kill -6 pid
   ```

A small core file is created in the directory where **start_iq** was run. If you can kill the server process in this way, skip to step 5.

3. If the server process still does not exit, capture **ps** output as in step 1. Retain the output from both times you run **ps** (before and after trying to kill the process). Then kill the process with a stronger signal:

   ```
   kill -9 pid
   ```

4. If this method does not cause the process to exit, capture yet another set of **ps** output, then restart your system.
5. Submit all **ps** output, the core file (if generated in step 2), and the stack trace in `stktrc-YYYYMMDD-HHNNSS_#.iq` to Technical Support.

On Windows systems:

1. Start the Task Manager by right-clicking the Task Bar and clicking **Task Manager**.
2. In the Processes tab, select **iqsrv16.exe**, then click the **End Process** button to stop the database server.
3. If necessary, restart Windows.

Database Connection Issues

You may encounter issues while attempting to connect to a database.

Cannot Connect to a Database

You may experience problems connecting to a database.

Possible Causes

- Data source is not defined, or is defined incorrectly. Try connecting again with the correct user ID and password.

  A data source is a set of connection parameters, stored in the registry (on Windows) or in a file (Windows and UNIX).
- An incorrect user name or password is specified.
- User may not have permission to use the database.
- You are connecting over TDS (for example, using jConnect) and the user ID or password is longer than 30 bytes. You see:

  ```
  Invalid user ID or password
  CT-LIBRARY error:
  ct_connect(): protocol specific layer:
  external error: The attempt to connect to the server failed.
  ```

- You provide an incorrect database file name. Try connecting again with the correct database file name.

  You must supply the **DBF** parameter and the database file name to connect when you use Interactive SQL and you have restored the database from backup while connected to `utility_db`.

- Database files may be missing. The files `dbname.db`, `dbname.iq`, and `dbname.iqmsg` (where `dbname` is the name of the database) must all exist.
- A limit on the number of connections or other DBA-defined login restrictions may be exceeded.
- You have run out of disk space. Check the SAP Sybase IQ message file for messages related to disk space.
- The server name specified is incorrect. Check the name of the server and try connecting again with the correct server name.
- The server machine name or address has changed.
- When connecting from a client for the first time and the server name is not specified, providing the wrong port number can cause a failure to connect to the database. The error messages returned are:

  ```
  Could not connect to the database.
  Database server not found.
  ```

  When connecting from Interactive SQL, ensure that the name in the Server Name field is spelled correctly, that the network options on the network tab are correct, and that the database server has been started. Either provide the server name when connecting, or use the correct port number. To determine the server name and the number of the port on which the server is listening, run **stop_iq** (UNIX), which displays this information.
- Port number may be out of correct range or in use by another process.
- If you receive the either the message:

  ```
  Unable to start — server not found
  ```

  or:

  ```
  Database server not running.
  ```

  when trying to start the client, the client cannot find the database server on the network. The connection string may be incorrect or the server name cache may contain incorrect or old connection information. For example, if the server is started with a different port number, even if the client application specifies the new port number at connect time, the connection information is still taken from the server name cache.
- You specified a character set in the CharSet connection parameter and tried to connect to a server that does not support that character set.

  Try reconnecting without specifying CharSet. If the client's local character set is unsupported by the server, the connection succeeds, but with a warning that the character set is not supported.

Action

If you suspect that you cannot connect because there is a problem with the database, look in the `dbname.iqmsg` file to determine where the problem occurred.

`Open Database Completed` indicates that the database opened without error and the problem is related to the clients connecting. If the message does not appear, then the database may have failed while opening or recovering.

Interactive SQL (dbisql) Issues

You may encounter issues while using **dbisql**.

Directories Remain After Exiting dbisql

This issue affects users of NFS file systems only.

Possible Cause

The IQTMP16 environment variable does not point to a local directory. Each client connection creates several directories and files in a temporary directory. SAP Sybase IQ deletes these files when the connection ends. If IQTMP16 does not point to a local directory, it cannot find the `.nfs*` files that NFS creates.

Action

Set IQTMP16 to a local directory and restart the server.

Resource Issues

Resource issues may include insufficient disk space, insufficient number of threads, thread stack overflow, and unused system resources.

Insufficient Disk Space

The SAP Sybase IQ server does not wait for additional space on an out-of-dbspace condition, but rolls back either the entire transaction or rolls back to a savepoint.

If there is not enough temporary or main dbspace available for a buffer or dbspace allocation request, the statement making the request rolls back.

At this point, the DBA can add more space to a dbspace using the **ALTER DBSPACE** or the **ALTER FILE** command. (You may choose to add files instead of dbspaces. A single dbspace can have multiple dbfiles.)

Warning! If SAP Sybase IQ holds certain system locks or is performing a checkpoint when you run out of disk space, you may not be able to add disk space. It is important for you to recognize when you are low on disk space, and to add a new dbspace before you run out of space.

Actions

- Check recent messages in the SAP Sybase IQ message log (`dbname.iqmsg`). An `out of space` message indicates that you must add another dbspace. The message in the SAP Sybase IQ message file indicates which dbspace has run out of space. If the problem occurs while you are inserting data, you probably need more room in the IQ main store. If the problem occurs during queries with large sort-merge joins, you probably need more room in the IQ temporary store.

  Check the SAP Sybase IQ message log for the following messages:

- If a buffer or dbspace allocation request fails because there is no space in the dbspace, this error message is logged in the `dbname.iqmsg` message file:

  ```
  You have run out of space in %2 DBSpace. %1
  ```

  ```
  [EMSG_OUT_OF_DBSPACE: SQL Code -1009170L,
  SQL State QSB66, Sybase Error Code 20223]
  ```

  where %2 is the name of the dbspace.
- If the entire transaction is rolled back on an out-of-dbspace condition, you see:

  ```
  %1 -- Transaction rolled back"
  ```

  ```
  [IQ_TRANSACTION_ROLLBACK: SQL Code -1285L,
  SQL State 40W09, Sybase Error Code 2973]
  ```

  where %1 is the error that caused the transaction to roll back, when encountered by the server during a critical operation.
- If a buffer allocation request finds a dirty buffer, but the buffer manager cannot flush the buffer due to an out-of-space condition, you see this message, and the current statement rolls back:

  ```
  %2: All buffer cache pages are in use, ask your
  DBA to increase the size of the buffer cache. %1
  ```

  ```
  [EMSG_BUFMAN_ALLSLOTSLOCKED: SQL Code -1009031L,
  SQL State QSA31, Sybase Error Code 20052]
  ```

  where %2 is the particular buffer cache throwing the exception.

- Try to connect to the database from a new connection. If this works, the database server is running, even though the query is waiting. Run **sp_iqstatus** to get more information.
- If you cannot connect to the database, check if SAP Sybase IQ is in an unusable state by monitoring the CPU usage for that processor. If the CPU usage does not change over a small time interval, then SAP Sybase IQ is probably not operational. If the CPU usage does change, SAP Sybase IQ is operational.
- Check the **sp_iqstatus** output for:

  ```
  Main IQ Blocks Used:,10188 of 12288,
  82%, Max Block#: 134840
  ```

  ```
  Temporary IQ Blocks Used:,163 of 6144,
  2%, Max Block#: 97
  ```

  If the percentage of blocks used is in the nineties, add more disk space with the **CREATE DBSPACE** command. In this example, 82% of the Main IQ Blocks and 2% of the Temporary IQ Blocks are used, indicating that more space will soon be needed in the IQ main store.
- If out-of-space conditions occur, or **sp_iqstatus** shows a high percentage of main blocks in use on a multiplex server, run **sp_iqversionuse** to find out which versions are being used and the amount of space that can be recovered by releasing versions.

Running out of Space During Checkpointing

Start in forced recovery mode and add space as soon as possible.

You must add a dbspace before any new checkpoints can succeed.

Effect of Checkpoints on Out-of-Disk Space Conditions

If SAP Sybase IQ has already run out of space when a checkpoint is requested, the **checkpoint** command fails with an error.

```
You have run out of space during the CHECKPOINT operation.
```

```
[EMSG_IQSTORE_OUTOFSPACE_CHECKPOINT:'QSB33', 1009133].
```

You must add a dbspace before any new checkpoints can succeed.

Adding Space If You Cannot Connect to a Server

If you run out of space during an operation and cannot add space because you cannot connect to the server, add space using the **CREATE DBSPACE** command.

1. Shut down the server using any of these methods:
   - On any platform, run **dbstop**.
   - On Windows, click the correct server icon on the Windows task bar to display the SAP Sybase IQ window, then click the **Shutdown** button.
   - On UNIX, run **stop_iq** or type `q` in the window where the server was started.
2. Restart the engine with the **start_iq** command.
3. Connect to the database.
4. Use the **CREATE DBSPACE** command to add space.
5. Re-run the operation that originally failed due to insufficient space.

Managing Dbspace Size

Growth of catalog files is normal and varies depending on application and catalog content. The size of the `.db` file does not affect performance, and free pages within the `.db` file are reused as necessary.

To minimize catalog file growth:

- Avoid using **IN SYSTEM** on **CREATE TABLE** statements.
- Issue **COMMIT** statements after running system stored procedures.
- Issue **COMMIT** statements after long-running transactions

If the catalog store cannot extend one of its files (`.tmp`, `.db`, or `.iqmsg`), SAP Sybase IQ returns `A dbspace has reached its maximum file size`. To prevent this problem:

- Periodically monitor space usage.
- Verify that there are no operating system file size limits (such as Sun Solaris **ulimit**) where the `.tmp`, `.db`, or `.iqmsg` files are located. The `.db` and `.tmp` files are typically in the main SAP Sybase IQ database directory. The `.tmp` file is located under `$IQTMP16/<servername>/tmp`, or if $IQTMP16 is not set, under `/tmp/.SQLAnywhere/<servername>/tmp`.

Adding the Wrong Type of Space

If the temporary dbspace runs out of space and you omit the **TEMPORARY** keyword from the **CREATE DBSPACE** command, you cannot create a temporary dbspace.

Instead, add the file in the existing temporary dbspace as IQ_SYSTEM_TEMP.

Fragmentation

SAP Sybase IQ provides control over fragmentation by taking advantage of even the smallest unused spaces.

However, fragmentation can still occur. If your database runs out of space, even though Mem Usage listed by **sp_iqstatus** or the `.iqmsg` file shows that the Main IQ Blocks Used value is less than 100%, it usually indicates that your database is fragmented,

Freeing Space

When a connection is out of space, you cannot free space by dropping tables or indexes in another connection; the out-of-space transaction sees those objects in its snapshot version.

Reserving Space for the Future

SAP Sybase IQ automatically reserves the minimum of 200MB and 50 percent of the size of the last dbspace.

To ensure that you have enough room to add new dbspaces if you run out of space in the future, set the database options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DBSPACE_MB` to values that are large enough to handle running out of space during a **COMMIT** or **CHECKPOINT**.

Monitoring Disk Space Usage

You can use an event handler to monitor disk space usage and notify you when available space is running low.

The first example in this section is especially useful for monitoring space during loads. You can enable the event handler before you start the load, and disable it after the load completes.

You can modify this sample event handler code to perform other types of monitoring.

```
-- This event handler sends email to the database
-- administrator whenever the IQ main DBSpace is more than
-- 95 percent full.

-- This event handler runs every minute. The event handler uses
```

```
-- sp_iqspaceused to sample the space usage. If the space is
-- more than 95 percent full, a file that contains the date and
-- time is created in the directory where iqsrv16 is
-- running. The file contents are then mailed to the database
-- administrator and the file is removed.
-- This event can be enabled before a load and be used
-- to monitor disk space usage during loading. The event can
-- then be disabled after the load.

create event out_of_space
schedule
start time '1:00AM' every 1 minutes
handler

begin
declare mt unsigned bigint;
declare mu unsigned bigint;
declare tt unsigned bigint;
declare tu unsigned bigint;

call sp_iqspaceused(mt, mu, tt, tu);

if mu*100/mt  > 95  then
  call xp_cmdshell('date > ./temp_m_file');
  call xp_cmdshell('mailx -s add_main_dbspace iqdba@iqdemo.com
    < ./temp_m_file');
  call xp_cmdshell('/bin/rm -rf ./temp_m_file');
end if;

if tu*100/tt  > 95  then
  call xp_cmdshell('date > ./temp_file');
  call xp_cmdshell('mailx -s add_temp_dbspace iqdba@iqdemo.com
    < ./temp_file');
  call xp_cmdshell('/bin/rm -rf ./temp_file');
end if;

end
```

The following code creates a timer-based event that monitors space usage to help avoid unexpected rollbacks, which may occur in out-of-space situations on operations without privileges. The `DBSpaceLogger` event is created in the sample `iqdemo` database.

```
CREATE EVENT DBSpaceLogger
SCHEDULE START TIME '00:00:01' EVERY 300 SECONDS
HANDLER
BEGIN
DECLARE DBSpaceName VARCHAR(128);
DECLARE Usage SMALLINT;
DECLARE cursor_1 CURSOR FOR
SELECT DBSpaceName, Usage
FROM sp_iqdbspace()
WHERE Usage > 0
ORDER BY Usage
FOR READ ONLY;
```

```
OPEN cursor_1;
idx1: LOOP
FETCH cursor_1 INTO DBSpaceName, Usage;
IF SQLCODE <> 0 THEN LEAVE idx1 END IF;
IF Usage >= 70 AND Usage < 80 THEN
call dbo.sp_iqlogtoiqmsg('Information: DBSpace' +
DBSpaceName + '''s usage is more than 70%');
ELSEIF Usage >= 80 AND Usage < 90 THEN
call dbo.sp_iqlogtoiqmsg('Warning: DBSpace ' +
DBSpaceName + '''s usage is more than 80%');
ELSEIF Usage >= 90 AND Usage < 100 THEN
call dbo.sp_iqlogtoiqmsg('Critical Warning: DBSpace
' + DBSpaceName + '''s usage is more than 90%');
END IF;
END LOOP;
CLOSE cursor_1;
END;
```

Insufficient Threads

The required number of server threads may not be available for your query.

Possible Cause

A client message similar to `Not enough server threads available for this query [-1010011] ['QXA11']` indicates that the query requires additional kernel threads for the IQ store.

Actions

- Wait for another query to finish and release the threads it is using. Then resubmit your query.
- Run **sp_iqconnection**. The column `IQThreads` contains the number of IQ threads currently assigned to the connection. This column can help you determine which connections are using the most resources. Some threads may be assigned but idle.
- If the condition persists, you may need to restart the server and specify additional IQ threads. Use the **-iqmt** server startup switch to increase the number of processing threads that SAP Sybase IQ can use.
  The default is 60 threads per CPU for the first 4 CPUs, and 50 threads per CPU for the remainder, with 3 more for system use, plus threads needed for database connections and background tasks. For example, on a system with 12 CPUs and 10 connections: `60*4 + 50*(numCPUs - 4) + numConnections + 6 = 656`. The minimum value is `numConnections + 3`. The total number of server threads cannot exceed 4096 on 64-bit platforms, or 2048 on 32-bit platforms.
- If the server runs out of threads, or if sufficient threads are not available to a connection during a restore, you may see the error `Ran out of threads. Start up server with more threads.` (SQLCODE -1012024). The **RESTORE** command attempts to allocate a “team” of threads for the restore operation. SAP Sybase IQ attempts to allocate at least one thread per backup device, plus two threads per CPU, plus one thread to the team. Make sure you have allocated enough threads on both a per-connection and

per-team basis, as well as to the server. Use the `MAX_IQ_THREADS_PER_CONNECTION` and `MAX_IQ_THREADS_PER_TEAM` database options.

Stack Overflow

You may experience problems if the thread stack overflows.

`AbortIfEndofStack` in the stack trace file (`stktrc-YYYYMMDD-HHNNSS_#.iq`), indicates that the thread stack has overflowed.

Possible Causes

- To avoid this problem, restart SAP Sybase IQ with the server parameter **-iqtss** set to 300 on 32-bit operating systems, or 500 on 64-bit operating systems. The server startup switch **-iqtss** specifies thread stack size in KB. If this is inadequate, raise the value of **-iqtss** by 72 until the problem is solved.
- If possible, identify the command that caused the error and forward it to Technical Support.

Unused Semaphores and Shared Memory Left After Abnormal Exit

Abnormal exits may leave unused semaphores and shared memory.

AIX, HP-UX, and Linux platforms use semaphores for communication between clients and servers on the same computer. Each client allocates one semaphore, as does each server. A client signals the server's semaphore when it has placed a packet for the server to read, and vice versa. The number of semaphores needed for a given system depends on how many local client applications connect via shared memory to the local server. If a client needs to allocate multiple semaphores for multiplex connections to one or more servers, it attempts to allocate all semaphores in the semaphore group.

Possible Causes

Killing processes on UNIX systems may result in semaphores or shared memory being left behind instead of being cleaned up automatically. To eliminate unneeded semaphores, periodically run the UNIX **ipcs** command to check the status of semaphores and shared memory.

The **ipcs -a** command lists the ID numbers, owners, and create times of semaphores and shared memory segments. When all SAP Sybase IQ instances are started by the same user (which is recommended), you can search the OWNER column for that user name. Identify shared memory segments and semaphores that are not being used.

Action

After verifying with the owner that these shared memory segments and semaphores are not in use, run the UNIX **ipcrm** command to remove them. Use the **-m** parameter to specify the memory segment ID and the **-s** command to specify the semaphore ID number, in the following format:

```
ipcrm -m mid1 -m mid2 ... -s sid1 -s sid2 ...
```

For example:

```
% ipcrm -m 40965 -s 5130 -s36682
```

Insufficient Buffers

If the resource manager determines that there is not enough cache to complete an operation, the operation is not started, and `Insufficient buffers` is returned.

Use one of these suggestions to resolve the issue:

- Increase the buffer cache size and re-run the operation.
- Reschedule the operation to a time when the server is not as busy and more buffer cache might be available.
- In a multiplex environment, move the workload to another node.

Processing Issues

Processing issues may be related to loads, queries, indexes, and table access.

Too Many Indexes on Table

Issues may occur when a table has too many indexes.

Possible Cause

A Microsoft Access user is trying to link to a table that has more than 32 indexes.

Action

Create a view that selects all the columns in the table, and link to the view instead of the base table.

Unexpectedly Long Loads or Queries

Long loads or queries may cause issues.

Possible Causes

- IQ buffer cache is too large, so the operating system is thrashing.
- IQ buffer cache is too small, so SAP Sybase IQ is thrashing because it cannot fit enough of the query data into the cache.
- You attempted to set IQ buffer cache sizes so that total memory requirements on your system exceed total system memory. Consequently, buffer caches have been therefore automatically reduced to their default sizes.
- User-defined functions or cross-database joins requiring CIS intervention.
- Missing HG or LF index on columns used in the **WHERE** clause and **GROUP BY** clause.

Action

Monitor paging to determine if thrashing is a problem.

- To monitor IQ paging, run the IQ buffer cache monitor.
- To monitor operating system paging, use the UNIX **vmstat** utility or other platform-specific tools, or the Windows Performance Monitor.

Reset your buffer sizes as needed.

You can also limit the amount of thrashing during a query execution that involves hash algorithms. Adjusting the HASH_THRASHING_PERCENT database option controls the percentage of hard disk I/Os allowed before the statement is rolled back and an error is returned.

The default value of HASH_THRASHING_PERCENT is 10%. Increasing the value permits more paging to disk before a rollback, and decreasing the value permits less paging before a rollback.

Queries involving hash algorithms that executed in earlier versions of SAP Sybase IQ may now be rolled back when the default HASH_THRASHING_PERCENT limit is reached, and you may see either of these messages:

```
Hash insert thrashing detected.
```

```
Hash find thrashing detected.
```
(SQLState QFA43, SQLCode -1001047)

To provide the query with the resources required for execution, perform one or more of these actions:

- Relax the paging restriction by increasing the value of HASH_THRASHING_PERCENT.
- Increase the size of the temporary cache (DBA only). Increasing the size of the temporary cache reduces the size of the main cache.
- Attempt to identify and alleviate why SAP Sybase IQ is incorrectly estimating one or more hash sizes for this statement.
- Decrease the value of the database option HASH_PINNABLE_CACHE_PERCENT.

To identify possible problems with a query, generate a query plan by running the query with the temporary database options QUERY_PLAN = 'ON' and QUERY _DETAIL = 'ON', then examine the estimates in the query plan. The option QUERY_PLAN_AFTER_RUN = 'ON' provides additional information, as the query plan is printed after the query has finished running. The generated query plan is in the message log file.

Load Fails on Number of Unique Values

The number of unique values in a query may cause issues.

Possible Cause

The following message in the log file indicates that you have more than 10000 unique values in a column with an **LF** index:

```
1009103: Number of unique values exceeded for index.
index_name_LF 10000
```

The Low_Fast index is optimized for 1000 unique values, but has an upper limit of 10000.

Action

Replace the **LF** index with an **HG** index. Issue a **DROP INDEX** statement to drop the **LF** index identified in the error message. For example:

```
DROP INDEX DBA.employee.emp_lname_LF
```

Then issue a **CREATE INDEX** statement to create the new **HG** index. For example:

```
CREATE HG INDEX ON DBA.employee (emp_lname)
```

Cannot Write to a Locked Table

Locked tables may cause issues.

Possible Causes

The following error message is reported when writing to an object to which another user already has write access. `Cannot open the requested object for write in the current transaction (TxnID1). Another user has write access in transaction TxnID2.`

Action

Use the **sp_iqlocks** stored procedure to identify users who are blocking other users from writing to a table. This procedure displays information about locks currently held in the database, including the connection and user ID that holds the lock, the table on which the lock is held, the type of lock, and a name to identify the lock.

The error message also includes the transaction ID of the user who is attempting to write (TxnID1) and the transaction ID of the user who is currently writing (TxnID2). For more detailed information about the transaction that has locked the table, run the **sp_iqtransaction** stored procedure.

Managing Write Lock Contention on a Table

High contention for write locks on a table used by multiple users can impact processing, if most of the transactions can obtain the lock.

This sample stored procedure shows one method of managing contention for a write lock on a table. This procedure does not eliminate the write lock contention on the table, but does manage the contention, so that transactions can obtain the write lock.

The sample stored procedure code manages lock contention on a table named `dbo.event`, which records events. The procedure returns the `event_id` to the caller. This table is in high contention for write locks. The stored procedure `dbo.log_event` records information in the table `dbo.event`. If an access error occurs, the error is captured, the hopeful writer sleeps for five seconds, and then attempts to write to the table again. The five second retry interval is usually long enough for the contention to be resolved, so the write lock on the `dbo.event` table is available.

You can modify this code to perform other similar tasks.

```
if exists (select 1
           from   sys.sysprocedure a
           join   sys.sysuserperm b on a.creator = b.user_id
          where  a.proc_name = 'log_event' and b.user_name = 'dbo')
then
   drop procedure dbo.log_event;
end if;

create procedure dbo.log_event(in @event varchar(255))
on exception resume
begin
    declare @event_id   bigint;
    declare @res        char(5);
    set @event_id=0;
    loop1: loop
        commit work;
        select  max(event_id)+1
            into    @event_id
            from    dbo.event;
        insert  dbo.event
            values (@event_id,@event,current timestamp,null,null);
        set @res=sqlstate;
        if @res = ' ' or(@res <> 'QDA29' and @res <> 'QDA11') then
            leave loop1
        end if;
        call dbo.sleep(5);
    end loop loop1;
    commit work;
    return @event_id
end
```

To prevent a critical update operation from failing, you may reserve write locks on all required tables in advance. For example, the following example reserves write locks on the tables `SalesOrders`, `Customers`, and `SalesOrderItems`, which are required for a hypothetical update:

```
BEGIN
WHILE TRUE LOOP
    LOCK TABLE SalesOrders, SalesOrderItems, Customers IN WRITE MODE
WAIT '30:00:00';
    If SQLCODE indicates that lock could not be acquired
    then
        SET status_msg = 'lock for required tables
        not yet acquired - retrying';
        Message to client status_msg;
    ELSE
        BREAK;
    ENDIF;
END LOOP; // Locks on SalesOrders, SalesOrderItems, Customers are
acquired
Update table SalesOrders …;
INSERT INTO SalesOrderItems …;
LOAD INTO Customers …;
COMMIT;
END;
```

Checkpoint Hints

The default values for checkpoint time and recovery time are sufficient and in most cases do not need to be changed.

The time between checkpoints defaults to 60 minutes.

You can adjust the time between checkpoints when you start your server by changing the **-gc** and **-gr** options in the **start_iq** command or in the `dbname.cfg` configuration file. The **-gc** switch specifies the number of minutes for the checkpoint timeout period. The **-gr** switch specifies the number of minutes for the maximum recovery time. The database engine uses both switches to calculate the checkpoint time.

For details on **start_iq** database options, see the *Utility Guide*.

Performance Issues

Certain settings may impact performance.

Slow Performance on a Multi-CPU or Hyperthreaded Machine

Multi-CPU or hyperthreaded machines may experience issues.

Possible Cause

SAP Sybase IQ runs most efficiently when it knows how many physical CPUs are available to it. On a machine with hyperthreads turned on, or where SAP Sybase IQ cannot access all of the available CPUs, SAP Sybase IQ creates too many threads and run less efficiently than it should.

Action

Start the server with **-iqnumbercpus** set to the number of CPUs available to SAP Sybase IQ, overriding the physical number of CPUs.

For details on **start_iq** database options, see the *Utility Guide*.

Troubleshooting Network Communications

Network software involves several different components, increasing the likelihood of issues requiring troubleshooting.

The primary source of assistance in network troubleshooting is the documentation and technical support for your network communications software, as provided by your network communications software vendor. However, you can follow best practices and use diagnostic tools to obtain information on various conditions.

Using Compatible Protocols

If you have more than one protocol stack installed on the client or server computer, ensure that the client and the database server are using the same protocol.

The `-x` command line switch for the server selects a list of protocols for the server to use, and the CommLinks connection parameter does the same for the client application.

You can use these options to ensure that each application is using the same protocol.

By default, both the database server and client library use all available protocol stacks. The server supports client requests on any active protocol, and the client searches for a server on all active protocols.

For more information about the `start_iq` database startup utility `-x` switch, see the *Utility Guide*.

Using Current Drivers

Ensure that you have the latest version of the NDIS or ODI driver for your network adapter, as appropriate.

You should be able to obtain current network adapter drivers from the manufacturer or supplier of the adapter card.

Network adapter manufacturers and suppliers make the latest versions of drivers for their cards available. Most card manufacturers have a Web site from which you can download the latest versions of NDIS and ODI drivers.

You may also be able to obtain a current network adapter driver from the provider of your networking software.

When you download Novell client software, ODI drivers for some network adapters are included, in addition to the Novell software that is used for all network adapters.

Powering Down Your Computer Between Restarts

Some network adapter boards do not reset cleanly when you restart the computer. When you are troubleshooting, turn the computer off, wait a few seconds, and then turn it back on.

Diagnosing the Protocol Stack Layer by Layer

If you are having problems getting your client application to communicate with a database server, ensure that the client and the database server are using compatible protocol stacks.

One way to isolate network communication problems is to work up the protocol stack, testing whether each level of communication is working properly.

If you can connect to the server computer, the data link layer is working, regardless of whether the connection is made using the same higher-layer protocols you will be using for SAP Sybase IQ.

For example, try to connect to a disk drive on the computer running the database server from the computer running the client application.

Once you have verified that the data link layer is working, next verify other applications using the same network and transport layers as SAP Sybase IQ are working.

Testing a TCP/IP Protocol Stack

If you are running under TCP/IP, there are several applications you can use to test the compatibility of the client computer and server computer TCP/IP protocol stack.

Using Ping to Test the IP Layer

Each IP layer has an associated address—a four-integer period-separated number (such as 191.72.109.12). **ping** takes as an argument an IP address and attempts to send a single packet to the named IP protocol stack.

First, determine if your own protocol stack is operating correctly by "pinging" your own computer. For example, if your IP address is 191.72.109.12, enter this command at the command prompt:

```
ping 191.72.109.12
```

Wait to see if the packets are routed. If they are, you see output similar to:

```
c:> ping 191.72.109.12
Pinging 191.72.109.12 with 32 bytes of data:
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
Reply from 191.72.109.12: bytes=32 time<.10ms TTL=32
...
```

If the ping works, it indicates that the computer can route packets to itself. This is reasonable assurance that the IP layer is set up correctly. Ask someone else running TCP/IP for his or her IP address, and try pinging that computer.

Before proceeding with additional diagnostics, ensure that you can ping the computer running the database server from the client computer.

Using Telnet to Test the TCP/IP Stack

To further test the TCP/IP stack, start a server application on one computer, and a client program on the other computer, and test whether they can communicate properly.

There are several applications commonly provided with TCP/IP implementations that can be used for this purpose. To use the **telnet** command to test the TCP/IP stack:

1. Start a Telnet server process (or *daemon*) on one machine. Check your TCP/IP software documentation for instructions. For a typical command line Telnet program, enter this at the command prompt:

   ```
   telnetd
   ```

2. Start the Telnet client process on the other machine, and see if you get a connection. Again, check your TCP/IP software documentation for instructions. Typically, enter an instruction similar to:

   ```
   telnet server_name
   ```

   where *server_name* is the name or IP address of the computer running the Telnet server process.

Establishing a Telnet connection between these two machines indicates that the protocol stack is stable and the client and server should be able to communicate using the TCP/IP link. If you cannot establish a Telnet connection, there is a problem. Before proceeding with additional diagnostics, ensure that your TCP/IP protocol stack is working correctly.

Diagnosing Wiring Problems

Faulty network wiring or connectors can cause problems that are difficult to isolate.

Try re-creating problems on a similar machine with the same configuration. If a problem occurs on only one machine, it may indicate a wiring or hardware problem.

For information on detecting wiring problems under NetWare, see your Novell NetWare manuals. The Novell LANalyzer program is useful for diagnosing wiring problems with Ethernet or TokenRing networks. Your NetWare authorized reseller can also supply you with the name of a Certified NetWare Engineer who can help diagnose and solve wiring problems.

Checking Common Network Communications Problems

Familiarize yourself with common network communications problems and their solutions.

"Unable to start — server not found" Message

The message `Unable to start — server not found` when trying to start the client, indicates that the client cannot find the database server on the network.

- The network configuration parameters of your network driver on the client machine may be different from those on the server machine. For example, two Ethernet adapter cards should use a common frame type. For Novell NetWare, the frame type is specified in the `net.cfg` file. In Windows, look for the frame type setting in the Control Panel Network Settings.
- Under the TCP/IP protocol, clients search for database servers by broadcasting a request. Such broadcasts typically do not pass through gateways, so any database server on a machine in another (sub)network, is not found. In this case, you must supply the host name of the machine on which the server is running using the `-x` server startup command line option. This is required to connect to NetWare servers over TCP.

- Your network drivers or wiring is not installed properly.
- The network configuration parameters of your network driver may be incompatible with SAP Sybase IQ multiuser support.

"Unable to initialize any communication links" Message

The message `Unable to initialize any communication links`, indicates that no link can be established.

The probable cause is that your network drivers have not been installed. The server and the client try to start communication links using all available protocols, unless you have specified otherwise using the `-x` server startup option. Check your network documentation to find out how to install the driver you need to use.

Diagnostic Tools

Several tools help you diagnose various conditions.

Restoring to a New Temporary File Topology

If temporary dbfiles cannot be opened or are damaged, you can restore the database to a different temporary file topology.

1. Start the utility server so it ignores all temporary IQ file definitions in the backed up database during the restore:

   ```
   start_iq -n utility_startup_svr -c 32m
   -x 'tcpip{port=1234}' -iqnotemp
   ```

2. Restore the database:

   ```
   RESTORE DATABASE 'iqdemo'
   FROM '/system1/IQ16/IQ-16_0/demo/backup/iqmain'
   ```

3. Restart the restored database using the **-iqnotemp** flag:
4. Drop all the files in IQ_SYSTEM_TEMP:

   ```
   ALTER DBSPACE IQ_SYSTEM_TEMP DROP FILE ALL
   ```

5. Restart the server without the **-iqnotemp** flag:
6. Add new temporary dbfiles to IQ_SYSTEM_TEMP:

The sp_iqstatus Stored Procedure

The **sp_iqstatus** stored procedure provides a variety of IQ status information.

Note: The following example shows output from the `iqdemo` sample database. The sample user dbspace `iq_main` may not be present in your own user-created databases.

The following output is from the **sp_iqstatus** stored procedure:

Sybase IQ (TM)	Copyright (c) 1992-2013 by Sybase, Inc. All rights reserved.
Version:	16.0.0.6552/110812/P/GA/Sun_Sparc/OS 5.10/64bit/2012-08-12 03:08:39
Time Now:	2012-09-13 10:33:19.979
Build Time:	2012-08-13 03:08:39
File Format:	23 on 03/18/1999
Server Mode:	IQ Multiplex Coordinator Server
Catalog Format:	2
Stored Procedure Revision:	1
Page Size:	131072/8192blksz/16bpp
Number of Main DB Files:	2
Main Store Out Of Space:	N
Number of Shared Temp DB Files:	0
Shared Temp Store Out Of Space:	N
Number of Local Temp DB Files:	1
Local Temp Store Out Of Space:	N
DB Blocks: 1-12800	IQ_SYSTEM_MAIN
DB Blocks: 1045440-1058239	iq_main
Local Temp Blocks: 1-3200	IQ_SYSTEM_TEMP
Create Time:	2013-08-17 11:31:03.313
Update Time:	2013-09-12 10:32:00.077
Main IQ Buffers:	510, 64Mb
Temporary IQ Buffers:	510, 64Mb
Main IQ Blocks Used:	8076 of 19200, 42%=63Mb, Max Block#: 1051107
Shared Temporary IQ Blocks Used:	0 of 0, 0%=0Mb, Max Block#: 0
Local Temporary IQ Blocks Used:	113 of 1600, 7%=0Mb, Max Block#: 834
Main Reserved Blocks Available:	6400 of 6400, 100%=50Mb
Shared Temporary Reserved Blocks Available:	0 of 0, 0%=0Mb
Local Temporary Reserved Blocks Available:	1600 of 1600, 100%=12Mb

IQ Dynamic Memory:	Current: 150mb, Max: 150mb
Main IQ Buffers:'	Used: 509, Locked: 0
Temporary IQ Buffers:'	Used: 8, Locked: 0
Main IQ I/O:'	I: L184357/P71 O: C18370/D25255/P20297 D: 5613 C:51.8
Temporary IQ I/O:'	I: L248471/P0 O: C22502/D25269/P4896 D: 22494 C:59.3
Other Versions:'	2 = 0Mb
Active Txn Versions:'	0 = C:0Mb/D:0Mb
Last Full Backup ID:'	0
Last Full Backup Time:'	
Last Backup ID:	0
Last Backup Type:	None
Last Backup Time:	
DB Updated:	1
Blocks in next ISF Backup:	0 Blocks: =0Mb
Blocks in next ISI Backup:	0 Blocks: =0Mb
Main Tlvlog Size:	Pages: 2, Recs: 413, Replays: 0/0
DB File Encryption Status:	OFF

Key to `Main IQ I/O` and `Temporary IQ I/O` output codes:

- `I:` Input
- `L:` Logical pages read (“Finds”)
- `P:` Physical pages read
- `O:` Output
- `C` Pages Created
- `D` Pages Dirtied
- `P:` Physically Written
- `D:` Pages Destroyed
- `C:`Compression Ratio

Check the following information:

- The lines `Main IQ Blocks Used` and `Temporary IQ Blocks Used` tell what portion of your dbspaces is in use. If the percentage of blocks in use (the middle statistic on these lines) is in the high nineties, add a dbspace.
- The `Main IQ Blocks Used` and `Temporary IQ Blocks Used` are calculated based on the line `DB Blocks` (Total Main IQ Blocks) minus `Main Reserved Blocks Available`and the line `Temp Blocks` (Total Temp IQ Blocks) minus `Temporary Reserved Blocks Available`, since the Reserved Blocks cannot be used for user operations.
- The lines `Main IQ Buffers` and `Temporary IQ Buffers` tell you the current sizes of your main and temp buffer caches.
- `Other Versions` shows other db versions and the total space consumed. These versions will eventually be dropped when they are no longer referenced or referencable by active transactions.
- `Active Txn Versions` shows the number of active write transactions and the amount of data they have created and destroyed. If these transactions commit, the “destroyed” data becomes an old version and eventually be dropped. If they roll back, the “created” data is freed.
- `Main Reserved Blocks Available` and `Temporary Reserved Blocks Available` show the amount of available reserved space.
- The lines `Main IQ I/O` and `Temporary IQ I/O` display I/O status in the same format as in the IQ message log.

Interpreting Notification Messages

By default, SAP Sybase IQ displays information about your database during insert and load operations in the IQ message log (`.iqmsg` file).

The statistics in these messages indicate when you need to perform maintenance and optimization tasks, such as adding more dbspaces. The messages also report on the progress of the load.

At the start of the insert is a description of the operation, such as:

```
In table 'tab2', the full width insert
of 2 columns will begin at record 1.
I. 02/11 13:28:14. 0000000002 Insert Started:
I. 02/11 13:28:14. 0000000002 tab2
I. 02/11 13:28:14. 0000000227 [20895]: Insert Pass 1
completed in 0 seconds.
I. 02/11 13:28:14. 0000000227 [20895]: Insert Pass 2
completed in 0 seconds.
I. 02/11 13:28:14. 0000000227 [20834]:
    1 records were inserted into 'tab2'.
```

Each time SAP Sybase IQ inserts the number of records specified in the **NOTIFY** load option, the server sends a message such as:

```
2010-05-27 13:03:49 0000000002
[20897]: 100000 Records, 2 Seconds
```

The first line shows how many rows SAP Sybase IQ has read so far and the number of seconds taken since the last notification message to read these additional rows. Even if SAP Sybase IQ reads the same number of rows each time, the amount of time varies depending on the data read (for example, how many data conversions are required). Reported time intervals smaller than 1 second are usually reported as “0 Secs”.

Memory Message

The memory message displays information about memory usage of the SAP Sybase IQ server.

This line in the IQ message log (`.iqmsg` file) displays memory usage information:

```
Mem: 469mb/M470
```

Table 24. Memory Usage Message

Item	Description
Mem: # mb	Current memory, in megabytes, being used by this SAP Sybase IQ server.
M# mb	The maximum number of megabytes used by this SAP Sybase IQ server since it was started.

IQ Main Store Blocks Message

The IQ main store blocks message displays information about use of the blocks and buffers in the IQ main store.

This line in the IQ message log (`.iqmsg` file) describes the permanent IQ main store:

```
Main      Blks: U63137/6%, Buffers: U12578/L7
```

Table 25. IQ Main Store Blocks Message

Item	Description
U#	Number of blocks in use.
#%	Percentage of database filled.
Buffers: U#	Number of buffers in use. Normally this is 100% because the buffer manager leaves buffers in memory until they are needed for some other data. In general, the buffers used and buffers locked numbers are meaningless, because IQ uses buffers as aggressively and efficiently as it can. This value grows to the maximum number of buffers that fit in the main buffer cache. The number increments whenever a buffer is allocated, but decrements only when a buffer is destroyed, not when it is unlocked or flushed. Objects in the temporary cache release their buffers when they are finished, but in the main cache, IQ may or may not destroy the buffers because as long as a buffer is unlocked, it is available for reuse, whether it is empty, contains data, or contains destroyed data.

Item	Description
L#	Number of locked buffers. A locked buffer is in use and cannot be removed from the cache. IQ locks buffers of some objects, such as hash objects, to keep them in memory. It locks buffers of other objects, such as sorts, depending on the workload and what it considers a fair share for that object. This number increments whenever you request a buffer. If you exceed the maximum while running a script, the command that exceeds fails and subsequent commands may complete incorrectly. Buffer locks do not use any memory. A locked buffer has a flag set in the in-memory structure and the flag exists whether or not the buffer is locked.

It is important that you recognize when the server is low on disk space and that you add a new dbspace before the server runs out of space. For an example of using an event handler to monitor disk space usage and to notify you when available space is low during a load, see *Monitoring Disk Space Usage*.

IQ Temporary Store Blocks Message

The IQ temporary store blocks message displays information about use of the blocks and buffers in the IQ temporary store.

This line in the IQ message log (`.iqmsg` file) describes the Temporary IQ Store:

```
Temporary Blks: U273/0%, Buffers: U1987/L1960
```

Table 26. Temporary IQ Store Blocks Message

Item	Description
U#	Number of blocks in use.
#%	Percentage of database filled.
Buffers: U#	Number of buffers in use. Normally, this is 100% because the buffer manager leaves buffers in memory until they are needed for some other data. In general, the buffers used and buffers locked numbers are meaningless, because IQ uses buffers as aggressively and efficiently as it can. Objects in the temporary cache release their buffers when they are finished.

Item	Description
L#	Number of locked buffers. A locked buffer is in use and cannot be removed from the cache. IQ locks buffers of some objects, such as hash objects, to keep them in memory. It locks buffers of other objects, such as sorts, depending on the workload and what it considers a fair share for that object. This number increments when you request a buffer. If you exceed the maximum while running a script, the command that exceeds it fails and subsequent commands may complete incorrectly. Buffer locks do not use any memory. A locked buffer has a flag set in the in-memory structure and the flag exists whether or not the buffer is locked.

It is important that you recognize when the server is low on disk space and adding a new dbspace before the server runs out of space is important. For an example of using an event handler to monitor disk space usage and to notify you when available space is low during a load, see *Monitoring disk space usage*.

Main Buffer Cache Activity Message

The main buffer cache activity message displays information about the IQ main store buffer cache.

This line in the IQ message log (`.iqmsg` file) displays information about the IQ main store buffer cache:

```
Main       I: L331224/P22 O: D25967/P7805 C:D0
```

Table 27. IQ Main Store Buffer Cache Message

Item	Description
Main: I: L#	Number of logical file reads.
P#	Number of physical file reads.
O: D#	Number of times a buffer was destroyed.
P#	Number of physical writes.
C: D#	Buffer manager data compression ratio. This is the total number of bytes eligible for compression, minus the number of bytes used after compression, divided by total number of bytes eligible for compression times 100. In other words, how much data has been compressed (what percentage it is of its uncompressed size). The larger the number, the better. Only certain data blocks are eligible for compression. Eligible blocks include indexes, (90-95% of a database) and sort sets. The sort reflects only data compression techniques used by the buffer manager. Other data compression may take place before data reaches the buffer manager, so the total data compression may be higher.

In general, assuming the buffer cache is full, you should have between 10 and 1000 logical reads per physical read. A lower value indicates excessive thrashing in the buffer manager.

More than 1000 times larger can indicate that you may have allocated too much memory to your buffer cache.

Temporary Buffer Cache Message

The temporary buffer cache activity message displays information about the IQ temporary store buffer cache.

This line in the IQ message log (`.iqmsg` file) displays information about the IQ temporary store buffer cache:

```
Temporary I: L25240/P8 O: D4749/P0 C:D0
```

Table 28. Temporary IQ Store Buffer Cache Message

Item	Description
Temporary: I: L#	Number of logical file reads.
P#	Number of physical file reads.
O: D#	Number of times a buffer was destroyed.
P#	Number of physical writes.
C: D#	Buffer manager data compression ratio. This is the total number of bytes eligible for compression, minus number of bytes used after compression, divided by the total number of bytes eligible for compression, times 100. In other words, how much data has been compressed (what percentage it is of its uncompressed size). The larger the number, the better. Only certain data blocks are eligible for compression. Eligible blocks include indexes, (90-95% of a database) and sort sets. The ratio reflects only data compression techniques used by the buffer manager. Other data compression may take place before data reaches the buffer manager, so the total data compression may be higher.

In general, assuming the buffer cache is full, you should have between 10 and 1000 logical reads per physical read. A lower value indicates excessive thrashing in the buffer manager. More than 1000 times larger can indicate that you may be overallocating memory to your buffer cache.

User Name, Connection Handle, and Connection ID

After the temporary buffer cache message, the connection handle, connection ID (SA connID), and user name are logged in the `.iqmsg` file once per database connection.

These lines in the IQ message log (`.iqmsg` file) display connection information:

```
2010-05-12 09:34:42 0000000002 Txn 173
2010-05-12 09:34:42 0000000002 Connect: 1550990889. SA connID: 1.
User: DBA.
```

The connection handle is the value shown by the **sa_conn_info** stored procedure.

Note: To correlate connection information in the **-zr** log file with that in the `.iqmsg` file, see *Correlating connection information between the .srvlog and .iqmsg files*.

The sp_iqcheckdb Stored Procedure

If you suspect problems in your database, try running the stored procedure **sp_iqcheckdb**.

This procedure reads every database page from disk into memory and performs various consistency checks. However, depending on the size of your database, the check can take a long time to run.

sp_iqdbstatistics displays the database statistics collected by the most recent execution of **sp_iqcheckdb**.

Checking Database and Server Startup Option Values

When diagnosing server startup, resource, or processing issues, you may need to check the current values of database options and server startup options.

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** also lists server startup options that have been changed from the default values.

When a DBA executes **sp_iqcheckoptions** he or she sees all options that are set on a permanent basis for all roles and users, and sees temporary options set for DBA. Non-DBA users see their own temporary options. All users see non-default server startup options.

The **sp_iqcheckoptions** stored procedure requires no parameters. In Interactive SQL, execute:

```
sp_iqcheckoptions
```

The system table DBO.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.

Finding the Currently Executing Statement

When diagnosing a problem, you may want to know what statement was executing when the problem occurred.

The **sp_iqcontext** stored procedure lists currently running statements, and identifies the user and connection that issued the statement. You can use this utility together with information provided by **sp_iqconnection**, the `.iqmsg` log, and the **-zr** server request log (`.srvlog`), as well as stack traces, to determine what was happening when a problem occurred.

To match `.iqmsg` log and the **-zr** server request log entries using connection information, correlate connection information between the `.srvlog` and `.iqmsg` files.

Logging Server Requests

To isolate some types of problems, especially problems with queries, log server requests.

You can enable request-level logging by either:

- Setting the **-zr** command line option when you start the server, or
- Calling the **sa_server_option** stored procedure, which overrides the current setting of the **-zr**.

Server requests are logged in `*.srvlog`. The **-zr** server startup option enables request-level logging of operations and sets the type of requests to log (SQL | HOSTVARS | PLAN | PROCEDURES | TRIGGERS | OTHER | BLOCKS | REPLACE | ALL | NONE). The **-zo** option redirects request-level logging information to a file separate from the regular log file. The **-zs** limits the size of this file.

Note: If the size of the query text being written to the log exceeds the specified limit, the query text is not truncated and is logged in its entirety.

Once the database server is started, you can adjust the request log settings to log more or less information using **sa_server_option**. These commands enable request logging of a limited set of requests and redirect the output to the file `sqllog.txt`:

```
call sa_server_option('RequestLogging','SQL');
    call sa_server_option('RequestLogFile',
                          'sqllog.txt')
```

To disable request-level logging, use:

```
call sa_server_option('RequestLogging','NO');
```

To view the current settings for the SQL log file and logging level, execute:

```
select property('RequestLogFile'), property('RequestLogging');
```

To match `.iqmsg` log and the **-zr** server request log (`.srvlog`) entries using connection information, correlate connection information between the `.srvlog` and `.iqmsg` files.

Note: SAP Sybase IQ version 15.1 modified the request log. Instead of fixed-format line prefixes, common information began being recorded as comma-delimited text. Where possible, times are recorded as "=" (meaning the same as the previous line) or +nnn (meaning nnn milliseconds after the previous line). Consequently, request logs are much smaller now than in versions earlier than SAP Sybase IQ 15.1.

In addition, more information is recorded in the request log. For queries, the information recorded is isolation level, number of rows fetched, and cursor type. For **INSERT**, **UPDATE**, and **DELETE** statements, the information recorded is number of rows affected and number of triggers fired.

Optionally, you can also choose to log statements executed within procedures and triggers.

You can select to record the short form of query plans in the request log. If procedure logging is enabled, plans for procedure statements are also recorded.

The following output shows an excerpt from the request log, when the server is started with the **-zr all** option. In this example, the user connects to the `iqdemo` database and executes `sp_iqstatus`.

There are several comma-separated fields in each line, and the first field indicates the time. Periodically, a full timestamp is output in the form:

```
MMdd hhmmss.sss
0523 095954.807,[,1000000001,sp_iq_mpx_init,16,iq utilities status 1
```

For lines after this line, for example, "`+13,C,1,UID=DBA`", the offset is from the previous line. In this case, "`+13`" means that about 13 milliseconds have passed since the last line. In some cases, "=" means approximately 0 milliseconds have elapsed since the last line.

Here is the excerpt from the request log:

```
0523 095954.807,[,1000000001,sp_iq_mpx_init,16,iq
utilities status 1
+2,],1000000001,sp_iq_mpx_init,16
+1,[,1000000001,sp_iq_mpx_init,62,message STRING('IQ
Server ',@@servername,'.') to console
+2,],1000000001,sp_iq_mpx_init,62
taj% pg iqdemo.sqllog
0523 095954.807,[,1000000001,sp_iq_mpx_init,16,iq
utilities status 1
+2,],1000000001,sp_iq_mpx_init,16
+1,[,1000000001,sp_iq_mpx_init,62,message STRING('IQ
Server ',@@servername,'.') to console
+2,],1000000001,sp_iq_mpx_init,62
0523 100510.344,<,1,CONNECT
+13,C,1,UID=DBA
+83,>,1,CONNECT,1
+1,<,1,PREPARE,SELECT @@version, if 'A'<>'a' then 1
else 0 endif, isnull(property('IsIQ'),'NO'),
isnull(connection_property('odbc_distinguish_char_and_
varchar'),'Off'),
isnull(connection_property('odbc_describe_binary_as_va
rbinary'),'Off'), connection_property('charset'),
db_property('charset')
+1,>,1,PREPARE,65536
=,<,1,EXEC,65536
+79,P,1,[S]DUMMY<seq>
=,>,1,EXEC
+1,<,1,DROP_STMT,65536
=,>,1,DROP_STMT
=,<,1,PREPARE,SET TEMPORARY OPTION time_format =
'hh:nn:ss';SET TEMPORARY OPTION timestamp_format =
'yyyy-mm-dd hh:nn:ss.ssssss';SET TEMPORARY OPTION
date_format = 'yyyy-mm-dd';SET TEMPORARY OPTION
date_order = 'ymd';SET TEMPORARY OPTION isolation_level
= 0;
+1,>,1,PREPARE,65537
+1,<,1,EXEC,65537
=,[,1,*batch*,1,set temporary option time_format =
'hh:nn:ss'
```

```
+11,],1,*batch*,1
=,[,1,*batch*,1,set temporary option timestamp_format =
'yyyy-mm-dd hh:nn:ss.ssssss'
+11,],1,*batch*,1
+1,[,1,*batch*,1,set temporary option date_format =
'yyyy-mm-dd'
+11,],1,*batch*,1
=,[,1,*batch*,1,set temporary option date_order = 'ymd'+11,],
1,*batch*,1
=,[,1,*batch*,1,set temporary option isolation_level = 0
+11,],1,*batch*,1
=,>,1,EXEC
```

Request Log File Analysis

Use the stored procedures **sa_get_request_profile** and **sa_get_request_times** to read the **-zr** log file and summarize the results.

sa_get_request_profile analyzes the request log to determine the execution times of similar statements, and summarizes the results in the global temporary table `satmp_request_profile`. For example:

```
call sa_get_request_profile('/sys1/users/jones/iqreqs1_zr.log');
select * from satmp_request_profile;
```

sa_get_request_times also analyzes the request log to determine statement execution times and summarizes the results in the global temporary table `satmp_request_time`. For example:

```
call sa_get_request_times('/sys1/users/jones/iqreqs1_zr.log');
select * from satmp_request_time;
```

For more information about request-level logging, see the start_iq **-zo** switch in *Utility Guide > start_iq Database Server Startup Utility*, and the **sa_server_option** system procedure in *Reference: Building Blocks, Tables, and Procedures*.

Request logging

Request logging logs individual requests received from, and responses sent to, an application. It is most useful for determining what the database server is being asked to do by the application.

Request logging is also a good starting point for performance analysis of a specific application when it is not obvious whether the database server or the client is at fault. You can use request logging to determine the specific request to the database server that might be responsible for problems.

Note: All the functionality and data provided by the request logging feature is also available using diagnostic tracing. Diagnostic tracing also offers additional features and data.

Logged information includes such things as timestamps, connection IDs, and request type. For queries, it also includes the isolation level, number of rows fetched, and cursor type. For

INSERT, UPDATE, and DELETE statements, it also includes the number of rows affected and number of triggers fired.

Note: The request log contains all statements with obfuscated sensitive information. The only case when sensitive information is not obfuscated is a statement with a parsing error.

You can use the -zr server option to turn on request logging when you start the database server. You can redirect the output to a request log file for further analysis using the -zo server option. The -zn and -zs option let you specify the number of request log files that are saved and the maximum size of request log files.

The sa_get_request_times system procedure reads a request log and populates a global temporary table (satmp_request_time) with statements from the log and their execution times. For INSERT/UPDATE/DELETE statements, the time recorded is the time when the statements were executed. For queries, the time recorded is the total elapsed time from PREPARE to DROP (describe/open/fetch/close). That means you need to be aware of any open cursors.

Analyze satmp_request_time for statements that could be candidates for improvements. Statements that are inexpensive, but frequently executed, may represent performance problems.

You can use sa_get_request_profile to call sa_get_request_times and summarize satmp_request_time into another global temporary table called satmp_request_profile. This procedure also groups statements together and provides the number of calls, execution times, and so on.

Warning! If the log is being analyzed using the `tracetime.pl` Perl script, the max_client_statements_cached option should be set to 0 to disable client statement caching while the request log is captured.

Examples

Output to the request log can be filtered to include only requests from a specific connection or from a specific database, using the sa_server_option system procedure. This can help reduce the size of the log when monitoring a database server with many active connections or multiple databases.

- **Filter according to a connection –** Use the following syntax:
  ```
  CALL sa_server_option( 'RequestFilterConn' , connection-id );
  ```
  You can obtain *connection-id* by executing `CALL sa_conn_info( )`.
- **Filter according to a database –** Use the following syntax:
  ```
  CALL sa_server_option( 'RequestFilterDB' , database-id );
  ```
  The *database-id* can be obtained by executing `SELECT CONNECTION_PROPERTY( 'DBNumber' )` when connected to that database. Filtering remains in effect until explicitly reset, or until the database server is shut down.

- **Reset filtering –** Use either of the following two statements to reset filtering either by connection or by database:

```
CALL sa_server_option( 'RequestFilterConn' , -1 );
```

```
CALL sa_server_option( 'RequestFilterDB' , -1 );
```

- **Output host variables to request logs –** To include host variable values in the request log:
  - use the -zr server option with a value of `hostvars`
  - execute the following:

```
CALL sa_server_option( 'RequestLogging' , 'hostvars' );
```

  The request log analysis procedure, sa_get_request_times, recognizes host variables in the log and adds them to the global temporary table satmp_request_hostvar.

Connection for Collecting Diagnostic Information

The database option `DEDICATED_TASK` lets the DBA dedicate a request handling task to handle requests from a single connection.

This pre-established connection allows you to gather information about the state of the database server if it becomes otherwise unresponsive. See the **DEDICATED_TASK** option in *Reference: Statements and Options*.

Diagnosing Communications Issues

If your server returns a communication error on startup, you may want to set the **-z** command line option when you start the server.

This switch provides diagnostic information on communications links at server startup. Information is logged to standard output from where the server started and in the `srvlog` file.

Reporting Problems to Technical Support

If you cannot resolve a problem using the manuals or online help, the designated person should contact SAP Technical Support or the SAP subsidiary in your area.

Each SAP installation that has purchased a support contract has one or more designated people who are authorized to contact SAP Technical Support.

Technical Support needs information about your SAP Sybase IQ environment to resolve your problem.

If you have issues with Sybase Control Center, see the see the Sybase Control Center for SAP Sybase IQ online help in SCC or at *http://sybooks.sybase.com/sybooks/sybooks.xhtml?prodID=10680*.

Collecting Diagnostic Information Using getiqinfo

SAP Sybase IQ includes a script for collecting information that SAP Technical Support needs to diagnose problems.

The **getiqinfo** script collects information about the operating system environment, the SAP Sybase IQ environment, and log files.

Run this script before reporting a problem to SAP Technical Support. By doing so, you can help SAP staff resolve your issue more quickly, with less effort on your part.

getiqinfo is not designed for troubleshooting SAP Sybase IQ installations and does not provide on-site troubleshooting facilities. This script executes successfully only when the SAP Sybase IQ environment is properly set up and the server is running.

Before You Run getiqinfo

Collect information before running the **getiqinfo** script.

Before running the script, know the:

- Location of the database file
- Full path of the configuration file used to start the server, if one is used
- Full path of the `.iqmsg` file, if the SAP Sybase IQ message file has been renamed

If possible, leave the SAP Sybase IQ server running, or start the server before running **getiqinfo**. This allows the script to collect internal database data that is available only when SAP Sybase IQ is running. The script does not automatically start the server.

The script runs with the same environment settings that are used to start the SAP Sybase IQ server. **getiqinfo** uses some IQ-specific environment variables to search for files.

The script places collected data in the current directory (where you start the program). Be sure you have enough space under that directory. The script does not prompt for an alternative, but you can modify the script to change the output location by resetting the *DEST_DIR* variable.

Running the getiqinfo Script

On UNIX platforms, **getiqinfo** is a shell script. On Windows platforms, `getiqinfo.bat` is a batch script in the `IQ-16_0\bin64` directory. The **getiqinfo** script is installed with IQ Server; it is unavailable with the Network Client.

The steps vary for UNIX and Windows platforms.

1. Start the script according to your platform:
   - At the UNIX command prompt, in the `IQ-16_0/bin64` directory, type:
     ```
     getiqinfo.sh
     ```
   - In Windows, select **Start** > **Run** > `<install_path>\IQ-16_0\bin64\getiqinfo.bat`.

2. As you are prompted, enter:
   - The directory of the database file. This is also the default location of the `.iqmsg` file, and the `stktrc*.iq` file on UNIX.
   - The base name of the database file (the file name without the `.db` suffix). This is also the default base name of the `.iqmsg` file.
   - Other directories to search for these files.
   - SAP Sybase IQ engine name (server name) and port number for this database server.
   - User ID and password of a user granted one of:
     - DROP CONNECTION system privilege
     - MONITOR system privilege
     - SERVER OPERATOR system privilege
   - The full path to the configuration file used to start the SAP Sybase IQ server, if one was used.
   - The full path to the output file in the **-zo** server option, if one was specified.

The program also directs you to send the listed files to SAP Sybase Technical Support.

Information Collected by getiqinfo

The **getiqinfo** script collects information.

- Type of hardware, amount of memory, CPU type, speed, number of CPUs
- Operating system (for example, Sun Solaris 2.10)
- Swap space size
- SAP Sybase IQ version and EBF level, and Anywhere version
- Stack trace file for the date and time this problem occurred, named `stktrc-YYYYMMDD-HHMMSS_#.iq`, in the directory where you started the database server. (UNIX and Linux platforms only)
- Command or query that produced the error
- Message log file, named `dbname.iqmsg`, located, by default, in the directory where you started the database server
- Query plan (recorded in `.iqmsg` file; see the note below)
- Server logs
  - For UNIX, `IQ-16_0/logfiles/<servername>.000n.stderr` and `IQ-16_0/logfiles/<servername>.000n.srvlog`
  - On Windows platforms, if needed, you must restart the server and manually collect a copy of the console window
- Startup and connection option settings, from the configuration file (by default, `dbname.cfg`)
- Database option settings and output from **sa_conn_properties** (if the server is still running)

The following information is not collected by **getiqinfo**, but may be requested by Technical Support:

- Connectivity protocol used (for example, ODBC, JDBC, TDS)
- Open Client version
- Configuration type (single user or multi user)
- Front-end tool used (for example, Brio Query)
- Schema and indexes for the database
- Output from **sp_iqcheckdb** procedure

Query plan detail is collected automatically by **getiqinfo** if the options below are set. You can also collect this information manually, by setting the options and re-running the command that produced the error.

```
SET TEMPORARY OPTION QUERY_PLAN = 'ON'
```

```
SET TEMPORARY OPTION QUERY_DETAIL = 'ON'
```

The query plan is in the message log file. The default values for these options are QUERY_PLAN = ON and QUERY_DETAIL = OFF.

If you have performance problems, set the following option:

```
SET TEMPORARY OPTION QUERY_PLAN_AFTER_RUN = 'ON'
```

Setting this option enables Technical Support to see which steps in the query processing used the time.

Correlating Connection Information Between the .srvlog and .iqmsg Files

Technical Support may ask you to set the **-zr** option on the **start_iq** command in your configuration file.

This server startup option sets the request logging level to track statements sent to the server. Parameters are ALL, NONE, or SQL. The option produces a log file named for the server with the suffix `.srvlog`.

In the *SAP Sybase IQ* message file `.iqmsg`, each connection to the server is identified by a connection handle. The `.iqmsg` message file records the errors, warnings, and tracing information for each connection.

To correlate the connection identifiers in the `.srvlog` and `.iqmsg` files to find relevant information.

1. In the `.iqmsg` file, locate a connection of interest. For example:

   ```
   Connect:  SA connHandle: 1000000061
   ```

   These lines show the `.iqmsg` log file contents for this connection:

   ```
   16:14:59. 0000000062 Connect:  SA connHandle: 1000000061
             SA connID: 31  IQ connID: 0000000062  User: DBA
   ```

```
03/17 16:15:00. 0000000062 Cmt 12064
03/17 16:15:00. 0000000062 PostCmt 0
03/17 16:15:00. 0000000000 Disconnect:  SA connHandle: 1000000061
                SA connID: 31  IQ connID: 0000000062  User: DBA
```

2. Isolate all of the lines for the connection by searching the `.srvlog` file for the number that follows "SA connHandle" in the `.iqmsg` file.

   For example, search the `.srvlog` file for "1000000061":

```
16:14:59. [,1000000061,sp_iqdbspace,48,select
str_replace(dbspaceName,'"',null) into dbspaceName_literal
03/17 16:14:59. P,1000000061,[S][0]DUMMY<seq>
03/17 16:14:59. ],1000000061,sp_iqdbspace,48
03/17 16:14:59. P,1000000061,[1]ISYSIQDBFILE<seq> JNL
dbf<ISYSDBFILE>
JNL ISYSDBSPACE<ISYSDBSPACE>
03/17 16:14:59. [,1000000061,sp_iqdbspace,58,execute immediate
with
quotes on   'iq utilities main into iq_dbspace_temp dbspace info
' || dbspaceName
03/17 16:14:59. P,1000000061,[S]INSERT ROWS
03/17 16:14:59. P,1000000061,[S]INSERT ROWS
03/17 16:14:59. P,1000000061,[S]INSERT ROWS
03/17 16:14:59. P,1000000061,[S]INSERT ROWS03/17 16:14:59. P,
1000000061,[S]INSERT ROWS
03/17 16:14:59. ],1000000061,sp_iqdbspace,58
03/17 16:14:59. [,1000000061,sp_iqdbspace,60,select
d.dbspace_name as DBSpaceName,   min(SegType) as DBSpaceType,...
03/17 16:15:00. ],1000000061,sp_iqdbspace,60
03/17 16:15:00. P,1000000061,Work[ Sort[ GrByH[ dbf<seq> JNL
ISYSIQDBSPACE<ISYSIQDBSPACE> JNL ISYSDBSPACE<ISYSDBSPACE> JH*
iq_dbspace_temp<seq> ] ] ] : ISYSIQPARTITIONCOLUMN<seq> :
idx<seq> : tab<seq>
03/17 16:15:00. [,1000000061,sp_iqdbspace,105,drop table
dbo.iq_dbspace_temp
03/17 16:15:00. ],1000000061,sp_iqdbspace,105
03/17 16:15:00. P,1000000061,[1]Work[ Sort[ sp_iqdbspace<call> ] ]
```

   The connection handle in this example is 1000000061.

Support Web Site

If you cannot resolve a problem, you may find additional help on the SAP Sybase IQ online support Web site, MySybase.

MySybase lets you search through closed support cases, latest software bulletins, and resolved and known problems, using a view customized for your needs. You can even open a Technical Support case online.

MySybase can be used from most Internet browsers. Point your Web browser to `Technical Documents at http://www.sybase.com/support/techdocs/` and click MySybase for information on how to sign up for and use this free service.

Checklist: Information for Technical Support

You can run the **getiqinfo** script to collect information.

Information Requested	Value
SAP Sybase IQ version (for example 16.0 GA or SP number)	
sp_iqlmconfig output	
Type of hardware	
Amount of memory	
Number of CPUs	
Operating system name and version (for example, Microsoft Windows 2008 Service Pack 1)	
Operating system patch level	
Front-end tool used (for example, Business Objects Crystal Reports)	
Connectivity protocol used (for example, ODBC, JDBC, TDS)	
Open Client version	
Configuration type (single node or multiplex)	
Message log file (`dbname.iqmsg`)	
Server log files (`server.nnnn.srvlog` and `server.nnnn.stderr`)	
Stack trace file (`stktrc-YYYYMMDD-HHNNSS_#.iq`)	
Command or query that produced the error	
Start up option settings	
Connect option settings	
Database option settings	
Schema and indexes for the database	
sp_iqstatus output	
Query plan: set options (Query_Plan, Query_Detail, Query_Plan_After_Run, Query_Plan_As_Html, Query_Plan_As_Html_Directory, Query_Timing), re-run command or query	

Information Requested	Value
Screen snapshot of the problem, if possible.	

Appendix: Connection and Communication Parameters Reference

Connection parameters establish and describe connections from client applications to a database.

Connection Parameters

Connection parameters are included in connection strings.

You can enter connection parameters in:

- An application's connection string
- An ODBC data source
- The Sybase IQ Connect dialog

After each parameter name, the short form appears in square brackets. You can use the short form as an abbreviation in connect commands.

The ODBC configuration dialog and the Sybase IQ Connect dialog for Windows operating systems share a common format. Some parameters correspond to check boxes and fields in these dialogs, while others can be entered in the text box on the **Advanced** tab.

Usage Notes

Connection parameters are case-insensitive.

The usage for each connection parameter describes the circumstances under which the parameter. Common usage entries include:

- Embedded databases – when SAP Sybase IQ is used as an embedded database, the connection starts a server and loads the database. When the application disconnects from the database, the database is unloaded and the server stops.
- Network servers – when SAP Sybase IQ is used as a network server, the client application must locate a server already running on the network and connect to a database.

You can use the **dbping** utility to test connection strings. For examples, see the *Utility Guide > dbping Database Administration Utility*.

Boolean (true or false) arguments are YES, ON, 1, or TRUE if true; or NO, OFF, 0, or FALSE if false.

You can obtain the connection parameters used by the interface library from these places, in order of precedence:

- Connection string

- SQLCONNECT environment variable
- Data sources

The server name must be composed of characters in the range 1 to 127 of the ASCII character set. There is no such limitation on other parameters.

These rules govern the priority of parameters:

- The entries in a connection string are read left to right. If the same parameter is specified more than once, the last one in the string applies.
- If a string contains a DSN or FILEDSN entry, the profile is read from the configuration file, and the entries from the file are used if they are not already set. For example, if a connection string contains a data source name and sets some of the parameters contained in the data source explicitly, then in case of conflict, the explicit parameters are used.

AppInfo Connection Parameter [Appinfo]

Helps administrators identify the origin of particular client connections from a database server.

Usage
Anywhere

Default
Empty string

Description
This connection parameter is sent to the database server from Embedded SQL, ODBC, or OLE DB clients, as well as **DBISQLC** on UNIX. It is not available from Open Client.

The parameter consists of a generated string that holds information about the client process, such as the IP address of the client machine, the operating system it is running on, and so on. The string is associated in the database server with the connection, and you can retrieve it using:

```
SELECT connection_property( 'AppInfo' )
```

Clients can also specify their own string, which is appended to the generated string. The AppInfo property string is a sequence of semicolon-delimited **key**=*value* pairs. The valid keys are as follows:

- API – DBLIB, ODBC, OLEDB, or ADO.NET (ODBC is returned of the iAnywhere JDBC driver).
- APPINFO – string entered, if you specified AppInfo in the connection string
- EXE – name of the client executable (Windows only)
- HOST – host name of the client machine
- IP – IP address of the client machine

- OS – operating system name and version number (for example, Sun Solaris 2.9)
- PID – process ID of the client
- THREAD – thread ID of the client
- VERSION – version of the connection protocol in use, including major and minor values, and a build number (for example 9.0.1.1549)
- TIMEZONEADJUSTMENT – number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.

If you specify a debug log file in your client connection parameters, the APPINFO string is added to the file.

Examples

- Connect to the demo database from Interactive SQL (the iAnywhere JDBC driver is used by default):

  ```
  dbisql nogui -c "uid=DBA;pwd=sql" -host MachineName -port 1234
  ```

  View the application information:

  ```
  SELECT connection_property('AppInfo')
  ```

  The result is as follows (in a single string):

  ```
  IP=12.345.67.899;
  HOST=MachineName;
  OSUSER=sybase;
  OS='SunOS 5.10 Generic_144489_04';
  EXE='/Sybase/IQ16/shared/JRE-7_0_1/bin/amd64/java';
  PID=0x52af;
  THREAD=0x14;
  VERSION=16.0.0.290;
  API=iAnywhereJDBC;
  TIMEZONEADJUSTMENT=-240
  ```

- Connect to the default database from Interactive SQL, appending your own information to the AppInfo property:

  ```
  isql -U DBA -P sql -S MachineName:1234 --appname MyISQL
  ```

  View the application information:

  ```
  SELECT connection_property('AppInfo')
  ```

  The result is as follows (in a single string):

  ```
  HOST=MachineName;
  PID=21155;
  EXE=MyISQL
  ```

AutoPreCommit Connection Parameter [AutoPreCommit]

Forces each statement to commit before execution.

Usage
ODBC

Default
No

Description
By default, statements issue a commit after execution. When AutoPreCommit = Yes commit statements are issued before each select statement, so that users can always see the latest version of all database objects.

Example
You can set the AutoPreCommit option to Yes to turn on commit before execution or No to turn it off. Set this option in the `.odbc.ini` file or on the Advanced tab of the Connect dialog.

For example, this causes each statement to commit before execution:

```
[Sample DSN]
DatabaseFile=c:\Program Files\Sybase\IQ-16_0\demo\iqdemo.db
AutoPreCommit=Y
UserID=DBA
Password=SQL
```

AutoStart Connection Parameter [Astart]

Prevents a database server from being started if no connection is found.

Usage
Anywhere

Default
Yes

Description
By default, if no server is found during a connection attempt, and a database file is specified, then a database server is started on the same machine. You can turn this behavior off by setting the AutoStart parameter to Off in the connection string.

Example
sThe following data source fragment prevents a database server from being started if no network server is located:

```
[My Demo Database]
DatabaseFile=c:\sybase\IQ-16_0\demo\iqdemo.db
Autostart=No
UserID=DBA
ENG=network_server
```

AutoStop Connection Parameter [Astop]

Prevents unloading of a database as soon as there are no more open connections to it.

Usage
Embedded databases

Default
Yes

Description
By default, any server that is started from a connection string is stopped when there are no more connections to it. Also, any database that is loaded from a connection string is unloaded as soon as there are no more connections to it. This behavior is equivalent to **Autostop**=Yes.

If you supply **Autostop**=No, any database that you start in that connection is not unloaded when there are no more connections to it. As a consequence, the database server is not shut down either.

The **AutoStop** parameter is used only if you are connecting to a database that is not currently running. It is ignored if the database is already loaded.

Example
The following Windows connection profile prevents the database from being unloaded when the connection is dropped:

```
[Sample Embedded Database]
DatabaseFile=c:\sybase\IQ-16_0\demo\iqdemo.db
Autostop=No
UserID=DBA
```

CharSet Connection Parameter [CS]

Specifies the character set to be used on this connection.

Usage
Anywhere

Default
The locale character set

Description
If you supply a value for CharSet, the specified character set is used for the current connection. CharSet=none disables character set conversion for the connection.

When unloading data, you can specify the character set using the CharSet connection parameter.

To avoid lossy character set conversions, avoid setting the CharSet connection parameter when using Unicode client APIs. Unicode client APIs include ADO.NET, OLE DB, and the iAnywhere JDBC driver. ODBC is also a Unicode client API when the wide (Unicode) functions are used.

CompressionThreshold (COMPTH) connection parameter

Increases or decreases the size limit at which packets are compressed.

Syntax

```
{ CompressionThreshold | COMPTH }=size[ k ]
```

Usage

Anywhere except TDS. Only applies to compressed connections.

Allowed values

- **size –** This integer specifies the size limit at which packets are compressed. The default value is in bytes, but you can use `k` to specify units of kilobytes. If both the client and database server specify different compression threshold settings, the client setting applies. The minimum supported value is 1 byte, and the maximum supported value is 32767 bytes. Values less than 80 bytes are not recommended.

Default

120

If no CompressionThreshold value is set, the compression threshold value is controlled by the setting on the server, which defaults to 120 bytes.

Remarks

Changing the compression threshold can help performance of a compressed connection by allowing you to only compress packets when compression will increase the speed at which the packets are transferred.

When compression is enabled, individual packets may or may not be compressed, depending on their size. For example, SAP Sybase IQ does not compress packets smaller than the compression threshold, even if communication compression is enabled. As well, small packets (less than about 100 bytes) usually do not compress at all. Since CPU time is required to compress packets, attempting to compress small packets could actually decrease performance.

Lowering the compression threshold value may improve performance on very slow networks, while raising the compression threshold may improve performance by reducing CPU. However, since lowering the compression threshold value will increase CPU usage on both the client and server, a performance analysis should be done to determine whether changing the compression threshold is beneficial.

Example

Connect, with a compression threshold of 100 bytes.

```
CompressionThreshold=100
```

CommBufferSize Connection Parameter [CBSize]

Sets the maximum size of communication packets, in bytes. Use k to specify units of kilobytes.

Syntax

{ CommBufferSize | CBSIZE }=size[k]

Usage

Anywhere

Allowed Values

- **size –** This integer specifies the maximum size of communication packets. The default value is in bytes, but you can use **k** to specify units of kilobytes. The minimum value of CommBufferSize is 500 bytes, and the maximum is 65535 bytes.

Default

If no CommBufferSize value is set, the CommBufferSize is controlled by the setting on the server, which defaults to 7300 bytes.

Remarks

The protocol stack sets the maximum size of a packet on a network. If you set the CommBufferSize to be larger than that permitted by your network, the communication packets are broken up by the network software. The default size is a multiple of the standard ethernet TCP/IP maximum packet size (1460 bytes).

A larger packet size may improve performance for multi-row fetches and fetches of larger rows, but it also increases memory usage for both the client and the server.

If CommBufferSize is not specified on the client, the connection uses the server's buffer size. If CommBufferSize is specified on the client, the connection uses the CommBufferSize value.

Using the -p database server option to set the CommBufferSize causes all clients that do not specify their own CommBufferSize to use the size specified by the -p database server option

Example

To set the buffer size to 1460 bytes:

```
...
CommBufferSize=1460
...
```

Alternatively, you can set this parameter by entering its value in the **Buffer size** text box of the **Network** tab of the connection window.

CommBufferSpace Connection Parameter [CBSpace]

Specifies the amount of space to allocate on startup for network buffers, in kilobytes.

Usage
Anywhere

Values
Integer

Default
10

Description
CommBufferSpace is a global setting, for all connections.

Examples
The following profile fragment instructs the network library to allocate 200 KB for network buffers on startup:

```
...
CBSpace=200
...
```

You can set this parameter by entering its value in the **Buffer space** text box of the **Network** tab of the connection window.

CommLinks Connection Parameter [Links]

Specifies client-side network communications links.

Usage
Anywhere

Values
String

Default
Use only the shared memory communications link to connect.

Description
If you specify `CommLinks=ALL,` the client searches for a server using all available communication protocols. Since there may be an impact on performance if you specify `CommLinks=ALL`, use this setting only when you do not know which protocol to use.

If you specify one or more protocols in the CommLinks (LINKS) connection parameter, the client uses the named communication protocols, in the order specified, to search for a network database server. A connection error appears and the connection attempt aborts if the connection fails to connect using a specified protocol, even if there are protocols remaining in the list to try.

If you do not specify a CommLinks (LINKS) connection parameter, the client searches for a server on the current machine only, and only using a shared memory connection. This is the default behavior, and is equivalent to `CommLinks=ShMem`. The shared memory protocol is used for communication between a client and server running under the same operating system on the same machine.

Available values of the CommLinks parameter are:

- SharedMemory (ShMem) – start the shared memory protocol for same-machine communication. This is the default setting. The client tries shared memory first if it appears in a list of protocols, regardless of the order in which protocols appear.
- ALL –attempt to connect using the shared memory protocol first, followed by all remaining and available communications protocols. Use this setting if you are unsure of which communication protocol(s) to use.
- TCPIP – start the TCP/IP communications link. TCP/IP is supported on all operating systems.

Each of these values can have additional network communications parameters supplied.

Some of the reasons you may want to use a specific protocol, instead of **ALL**:

- The network library starts slightly faster if you do not start unnecessary network links.
- Connecting to the database may be faster.
- You must specify the link explicitly to tune the broadcast behavior of a particular protocol by providing additional network communications parameters.

Additional network communications parameters may be provided for each link, to tune the broadcast behavior of the link.

The **CommLinks** parameter corresponds to the database server **-x iqsrv16** server option. By default, the network server starts all available protocols, which is equivalent to **-x ALL**.

Examples

- The following connection string fragment starts the TCP/IP protocol only:

  ```
  CommLinks=tcpip
  ```
- The following connection string fragment starts the shared memory protocol and searches for the database server over shared memory. If the search fails, it then starts the TCP/IP port searching for the host **kangaroo** in addition to servers on the immediate TCP/IP network.

  ```
  CommLinks=tcpip(HOST=kangaroo),shmem
  ```

ConnectionName Connection Parameter [CON]

Names a connection to make switching to it easier in multi connection applications.

Usage
Not available for ODBC

Default
No connection name

Description
An optional parameter, providing a name for the particular connection you are making. You may leave this unspecified unless you are going to establish more than one connection, and switch between them.

The Connection Name is not the same as the data source name.

Examples
Connect, naming the connection FirstCon:

```
CON=FirstCon
```

ConnectionPool Connection Parameter [CPOOL]

Controls the behavior of client connection pooling.

Syntax
ConnectionPool={ *NO | YES [([Timeout=timeout-sec;] [MaxCached=max-cached-conn])]* **}**

Usage
All platforms except non-threaded Unix clients.

Allowed Values

- **timeout-sec –** The idle timeout period, in seconds, of the connection pool. The default value is 60 seconds. Cached connections that are not reused within the time specified by the timeout-sec value are disconnected and are no longer available for reuse.
- **max-cached-conn –** The maximum number of cached connections from each application. The default value is five connections. A connection is cached if it is disconnected and the maximum number of connections specified by the max-cached-conn value has not been reached. The connection is reinitialized, and the cached connection remains connected to the database server even though the application has disconnected it.

Default
YES

Remarks
Connection pooling may improve the performance of applications that make multiple, brief connections to the database server. When a connection is disconnected it is automatically cached and may be reused when the application reconnects. For a connection to be pooled, the connection name can be different, but all other connection parameters must be identical.

DatabaseFile Connection Parameter [DBF]

Specifies the target database file for a load or connection, when starting a database.

To connect to an already running database, use the **DatabaseName (DSN)** parameter.

SAP Sybase IQ requires the DBF parameter and the database file name to connect under special circumstances.

Usage
Embedded databases

Values
String

Default
None

Description
DBF loads and connects to a specific database file that is not already running on a database server.

- If a database is loaded with a name that is the same as the DatabaseFile parameter, but without the `.db` extension, the connection is made to that database instead.
- If the file name does not include an extension, a file of name `.db` is looked for.
- The path of the file is relative to the working directory of the database server. If you start the server from the command prompt, the working directory is the directory you are in when you enter the command. If you start the server from an icon or shortcut, it is the working directory specified in the icon or shortcut. It is recommended that you supply a complete path and file name.
- If you specify both the database file and the database name, the database file is ignored and is not used to try to connect to a database that is already running.

You can also use UNC file names.

Warning! The database file must be on the same machine as the database server. Managing a database file that is located on a network drive might lead to file corruption.

Example
To load and connect to the demo database (installed in directory `C:\Program Files\Sybase\IQ-16_0\demo` on Windows), use the following DBF parameter:

```
DBF=C:\Program Files\Sybase\IQ-16_0\demo\iqdemo.db
```

DatabaseName Connection Parameter [DBN]

Specifies a loaded database to which a connection needs to be made. Use when connecting to a database that is already running.

To connect to a database that is not already running, use the **DatabaseFile** (**DBF**) parameter.

Usage
Running network servers

Values
String

Default
None

Description
When a database is started on a server, it is assigned a database name, either by the administrator using the -n option, or by the server using the base of the file name with the extension and path removed.

If the database you want to connect to is already running, specify the database name rather than the database file.

A connection occurs only if the name of the running database matches the name that is specified in the DatabaseName (DBN) parameter.

Note: If you specify both the database name and database file, the database file is ignored and is not used to try to connect to a database that is already running.

Examples

- To start a database file named `cities.db` and rename the database Kitchener, you can use:
  ```
  start_iq cities.db -n Kitchener
  ```
- Assuming you have run the above command, you can successfully connect to the running database named Kitchener:
  ```
  DBN=Kitchener
  ```
- Alternatively:
  ```
  DBN=Kitchener;DBF=cities.db
  ```

- However, this fails to connect to the database named Kitchener:

```
DBF=cities.db
```

DatabaseSwitches Connection Parameter [DBS]

Provides database-specific switches when starting a database.

Usage
Connecting to a server when the database is not loaded.

Default
No switches

Description
Supply **DatabaseSwitches** only if you are connecting to a database that is not currently running. When the server starts the database specified by **DatabaseFile**, the server uses the supplied **DatabaseSwitches** as command line options to determine startup options for the database.

Only database switches can be supplied using this parameter. Server switches must be supplied using the START connection parameter.

Examples
The following UNIX command, entered all on one line, connects to the default database server, loads the database file IQ-16_0/demo/iqdemo.db (DBF parameter), names it as my_db (DBS parameter) and connects to the database of that name (DBN parameter):

```
dbcollat -c "uid=DBA;pwd=SQL;dbf=/IQ-16_0/demo/iqdemo.db;
dbn=my_db;dbs=-n my_db" /tmp/temp.col
```

DataSourceName Connection Parameter [DSN]

Tells the ODBC driver manager or Embedded SQL library where to look in the .odbc.ini file (on UNIX), or odbc.ini file or registry (on Windows), to find ODBC data source information.

Usage
Anywhere

Default
None

Description
It is common practice for ODBC applications to send only a data source name to ODBC. The ODBC driver manager and ODBC driver locate the data source, which contains the remainder

of the connection parameters. In SAP Sybase IQ, Embedded SQL applications can also use ODBC data sources to store connection parameters.

Example
This parameter uses a data source name:

```
DSN=Dynamo Demo
```

DBKEY Connection Parameter [DBKEY]

Starts an encrypted database with a connect request.

Usage
On database startup. Not used when connecting to a running database.

Default
By default, databases are not encrypted.

Description
You must specify this parameter when you start an encrypted database with a connect request.

Examples
The following profile fragment uses the encryption key “is!seCret” to connect to a strongly encrypted database named `marvin.db` that is already running:

```
...
UID=dba;PWD=sql;DBF=marvin.db;DBKEY='is!seCret'

...
```

DisableMultiRowFetch Connection Parameter [DMRF]

Turns off multi record fetches across the network.

Usage
Anywhere

Default
No

By default, when the database server gets a simple fetch request, the application asks for extra rows. You can disable this behavior by setting this parameter to ON.

Setting the **DisableMultiRowFetch** parameter to ON is equivalent to setting the PREFETCH option to OFF.

Example
This connection string fragment prevents prefetching:

```
DMRF=Yes
```

EngineName Connection Parameter [ENG]

Specifies a running database server to which you want to connect. This is a synonym for **ServerName**.

Usage

Network servers

Values

String

Default

The default local database server

Description

To connect to a network server, supply an **EngineName**. In the Connect dialog, and in the ODBC Administrator, this is the **Server Name** field.

The server name is interpreted according to the character set of the client machine. Multi byte characters are not recommended in server names.

Names must be a valid identifier. Long server names are truncated to different lengths depending on the protocol.

For database server naming restrictions, see the *Utility Guide*.

Protocol	Truncation Length
UNIX shared memory	31 bytes
Non-UNIX shared memory	40 bytes
TCP/IP	40 bytes

Examples

Connect to a server named **Guelph**:

```
ENG=Guelph
```

EncryptedPassword Connection Parameter [ENP]

Provides a password, stored in an encrypted fashion in a data source.

Usage

Anywhere (DSN and FILEDSN connection parameters do not support verbose form of keyword)

Values
String

Default
None

Description
Data sources are stored on disk as a file or in the registry. Storing passwords on disk may present a security problem. For this reason, when you enter a password into a data source, it is stored in an encrypted form.

If you specify both **Password** and **EncryptedPassword**, Password takes precedence.

Encryption Connection Parameter [ENC]

Encrypts packets sent between the client application and the server.

Usage
For **RSA_TLS**, TCP/IP only

For **NONE** or **SIMPLE**, anywhere

Values
String

Default
NONE

If an Encryption value is not set, encryption is controlled by the setting on the server, which defaults to no encryption.

Description
You can use this parameter if you are concerned about the security of network packets. Encryption marginally affects performance. The **Encryption (ENC)** connection parameter accepts these arguments:

- **None** accepts communication packets that are unencrypted. This value is equivalent to NO in earlier versions of SAP Sybase IQ.
- **Simple** accepts communication packets that are encrypted with simple encryption supported on all platforms and on pre-12.6 versions of SAP Sybase IQ. This value is equivalent to YES in earlier versions of SAP Sybase IQ.
- **RSA_TLS** accepts communication packets that are encrypted using RSA encryption technology. To use this type of encryption, both the server and the client must be operating on Solaris, Linux, and all supported Windows operating systems, and the connection must be over the TCP/IP port. UNIX platforms, except for Solaris and Linux, do not recognize the client or server **RSA_TLS** parameter. To authenticate the server, the software verifies

that the server's certificate values match any values you supply about the client using the following arguments:

- **trusted_certificates** specify the certificate file the client uses to authenticate the server.
- **certificate_company** specifies the value for the organization field. The server's value and the client's value must match.
- **certificate_unit** specifies the value for the organization unit field. The server's value and the client's value must match.
- **certificate_name** specifies the certificate's common name. The server's value and the client's value must match.

Warning! Use the sample certificate only for testing purposes. It provides no security in deployed situations because it and the corresponding password are widely distributed with SAP Sybase IQ software. To protect your system, create your own certificate.

You can use the **connection_property** system function to retrieve the encryption settings for the current connection.

Examples

- The following connection string fragment connects to a database server myeng with a TCP/IP link, using Certicom encryption and the sample trusted certificate:

```
"ENG=myeng; LINKS=tcpip; Encryption=TLS
(tls_type=ECC;trusted_certificate=sample.crt)"
```

- The following connection string fragment connects to a database server myeng with a TCP/IP link, using RSA encryption and the sample trusted certificate:

```
"ENG=myeng; LINKS=tcpip;
Encryption=TLS(tls_type=RSA;trusted_certificate=/Sybase/IQ/
certificate_authority_sample.pem)"
```

Escape Connection Parameter [ESCAPE]

Allows you to specify the escape character used in the LIKE clause of SQL statements generated by the ODBC driver when returning a list of tables or column.

Description

The **Escape** parameter is supported only by SAP Sybase IQ.

By default, the ODBC driver uses the tilde (~) as an escape character, but some applications assume that the escape character is a backslash (\). For example, the default escape character for characters stored as hexadecimal codes and symbols is a backslash (\), so `\x0A` represents the line feed character.

Examples

To use the exclamation mark as the escape character, enter:

```
ESCAPE='!'
```

FileDataSourceName Connection Parameter [FileDSN]

Tells the client library that there is an ODBC file data source holding information about the database to which you want to connect.

Usage
Anywhere

Values
String

Default
None

Description
File data sources hold the same information as ODBC data sources stored in a registry. File data sources can be easily distributed to end users, so that connection information does not have to be reconstructed on each machine.

Both ODBC and Embedded SQL applications can use file data sources.

Examples
This is a data source description held in a file data source:

```
[Sample File Data Source]
ENG=iqdemo
DBA=DBA
PWD=SQL
```

Idle Connection Parameter [IDLE]

The Idle Connection Parameter determines when a connection times out.

Function
Specifies the idle timeout period of the connection.

Usage
Anywhere except with Tabular Data Stream™ (TDS) and Shared Memory connections. Shared Memory and TDS connections (including jConnect) ignore the SAP Sybase IQ **Idle** (**IDLE**) connection parameter.

Values
Integer

Default
Value of **-ti**

Description
The **Idle** (**IDLE**) connection parameter applies only to the current connection. You can have multiple connections on the same server that are set to different timeout values.

If no connection idle timeout value is set, the idle timeout value is controlled by the setting on the server, which defaults to 4400 minutes when set by **start_iq**. In case of a conflict between timeout values, the connection timeout value supersedes any server timeout value whether specified or unspecified.

Optionally, the relevant server command line parameters can be included for both Idle and Liveness Timeout (**-ti** and **-tl** respectively).

See Also
-ti iqsrv16 server option in *Utility Guide > start_iq Database Server Startup Utility*.

Example
This connection string fragment sets the timeout value for this connection to 10 minutes:

```
"ENG=myeng;LINKS=tcpip;IDLE=10"
```

Integrated Connection Parameter [INT]

Uses the integrated login facility.

Usage
Anywhere

Values
YES, **NO**

Default
NO

Description
The **Integrated** parameter has the following settings:

- An integrated login is attempted. If the connection attempt fails and the LOGIN_MODE option is set to Mixed (deprecated), a standard login is attempted.
- This is the default setting. No integrated login is attempted.

For a client application to use an integrated login, the server must be running with the LOGIN_MODE database option set to Mixed (deprecated) or Integrated.

Examples
This data source fragment uses an integrated login:

```
INT=yes
```

Language Connection Parameter [LANG]

Specifies the language of the connection.

Usage
Anywhere

Values
The two-letter combination representing a language. For example, setting **LANG=DE** sets the default language to German.

Default
The language specified by (in order) the IQLANG environment variable or the installer.

Description
This connection parameter establishes the language for the connection. Any errors or warnings from the server are delivered in the specified language, assuming that the server supports the language.

If no language is specified, the default language is used.

This connection parameter affects only the connection. Messages returned from SQL Anywhere tools and utilities appear in the default language, while messages returned from the server appear in the connection's language.

LazyClose Connection Parameter [LCLOSE]

Causes the `CLOSE` *cursor-name* database request to queue and be sent to the server with the next database request. This eliminates a network request each time a cursor is closed.

Usage
Anywhere

Values
YES, **NO**

Default
NO

Description
When this parameter is enabled, cursors are not actually closed until the next database request. Any isolation level 1 cursor stability locks still apply to the cursor while the `CLOSE` *cursor-name* database request is queued.

Enabling this option can improve performance if:

- Your network exhibits poor latency
- Your application sends many cursor open and close requests

In rare circumstances, canceling the next request after the `CLOSE` *cursor-name* database request can leave the cursor in a state where it appears to be closed on the client side, but is not actually closed on the server side. Subsequent attempts to open another cursor with the same name fail. Do not use LazyClose if your application cancels requests frequently.

LivenessTimeout Connection Parameter [LTO]

Controls the termination of connections when they are no longer intact.

Usage

Network server on TCP/IP communications protocols.

All platforms except non threaded UNIX applications.

Values

Integer (in seconds)

Default

120

If no **LivenessTimeout** value is set, the liveness timeout is controlled by the setting on the server, which defaults to 120 seconds.

Description

A liveness packet is periodically sent across a client/server TCP/IP communications protocol to confirm that a connection is intact. If the client runs for the liveness timeout period without detecting a liveness request or response packet, communication is severed.

Liveness packets are sent when a connection has not sent any packets for between one third and two thirds of the **LivenessTimeout** value.

When communication is severed, the client machine forgets the address of the server. It looks the address up next time there is a connection to the server from that machine, dropping all current connections to that server.

When there are more than 200 connections to a server, the server automatically calculates a higher **LivenessTimeout** value based on the stated **LivenessTimeout** value. This enables the server to more efficiently handle a large number of connections.

Alternatively, you can set this parameter by entering its value in the LivenessTimeout text box of the Network tab of the ODBC Configuration dialog.

Example

This sets a Liveness timeout value of 60 seconds:

```
LTO=60
```

LogFile Connection Parameter [LOG]

Sends client error messages and debugging messages to a file.

Usage
Anywhere

Values
String

Description
To save client error messages and debugging messages in a file, use the **LogFile** (**LOG**) parameter.

If the file name includes a path, it is relative to the current working directory of the client application.

The LogFile (LOG) connection parameter is connection-specific, so from a single application you can set different LogFile arguments for different connections.

Examples
This command line fragment specifies that client messages for this connection should be sent to the file `error.log` in the current working directory for the client:

```
...
LogFile=error.log
...
```

LogicalServer Connection Parameter [LS]

Specifies the target logical server for a connection.

Usage
Anywhere

Values
String

Description
Use this parameter to explicitly specify the target logical server, enabling SAP Sybase IQ to redirect a connection from one multiplex node to another.

If LogicalServer is unspecified in the connection string, the setting of default logical server in the user's login policy serves as the logical server context.

LoginRedirection Connection Parameter [REDIRECT]

Controls login redirection at the connection level.

Usage
Login redirection at the logical server level must be enabled for this parameter to take effect.

Values
[**ON** | **OFF**]

Default
ON

Description
Certain applications may need to target specific nodes within a logical server.

When you specify **REDIRECT=OFF**, no login redirection takes place for that connection. The connection fails if the node of the initial connection cannot satisfy the connection requirements of target logical server and requested server role.

For example, you could disable login redirection at the connection level to target nodes within the logical server that contain data in SQL Anywhere tables, which is not shared between nodes.

NewPassword (NEWPWD) connection parameter

Allows users to change passwords, even if they have expired, without DBA intervention.

Syntax

```
{ NewPassword | NEWPWD }={ password-string | * }
```

Usage
The client library prompting for a new password is only supported on Microsoft Windows.

Allowed values

- **password-string –** If the user provides a new password, the database server authenticates the user ID and password and attempts to change the password before the login_procedure option is called. This process allows the user to change an expired password without the involvement of a DBA. If you have set the verify_password_function option, the new password is verified. If you are authenticating with an Integrated or Kerberos login, the original password is not validated and the database server ignores the new password value and the password is not changed.
- *** –** On Microsoft Windows, if you use the special value `*`, the client library prompts for a new password during a connection attempt only if the existing password has expired. The user must provide their existing password, provide their new password, and confirm their

new password. When the user completes the fields and clicks **OK**, the old password is authenticated and the database server attempts to change the password. If you have set the verify_password_function option, the new password is verified. The process of verifying if a user's password has expired, prompting for a password, and authenticating and changing the password occurs with a single connect call to the client library.

Default

The password is not changed, and the client library does not prompt for a new password.

Remarks

This connection parameter is very effective when you implement a login policy using the password_life_time or password_expiry_on_next_login options. Alternatively, you can implement a password expiry policy by having the login_procedure signal the `Password has expired` error.

A user receives a `Password has expired error` if their environment does not support password prompting. In a Microsoft Windows environment, the prompt window might not correctly prevent interaction with the calling application's window (it may not be modal or have the correct parent window) if the calling application has multiple top-level windows or if the application's top level windows are minimized.

In a Windows environment, if you use the ODBC SQLDriverConnect function and the DriverCompletion argument is anything other than SQL_DRIVER_NOPROMPT, the connection prompts for a new password if the password has expired. The connection might prompt for a new password in OLE DB when the DBPROP_INIT_PROMPT property is anything other than DBPROMPT_NOPROMPT. Both cases function as if the NewPassword=* connection parameter was specified.

- User IDs cannot:

  begin with white space, single quotes, or double quotes
  end with white space
  contain semicolons
- Passwords are case-sensitive and they cannot:

  begin with white space, single quotes, or double quotes
  end with white space
  contain semicolons
  be longer than 255 bytes in length

Example

The following connection string changes the password of user Test1 when they connect:

```
"UID=Test1;PWD=welcome;NEWPWD=hello"
```

In a Windows environment, the following connection string prompts the user Test1 for a new password when the existing password expires:

```
"UID=Test1;PWD=welcome;NEWPWD=*"
```

NodeType Connection Parameter

Specifies the server role for a logical server member node.

Usage
Anywhere

Values
{ **READER** | **WRITER** | **ANY** }

Default
ANY

Description
A logical server can have both reader and writer nodes as members. This parameter allows an application to connect to a member node with a specific role. You can specify the desired role as **NODETYPE = {READER | WRITER | ANY}**

- **READER** – applications that execute user-defined functions may need to connect to a member node with the **READER** role.
- **WRITER** – applications that execute statements (such as **INSERT**, **UPDATE**, and **DELETE**) may need to connect to a member nodewith the **WRITER** role.
- **ANY** – most read-only applications (that do not execute any UDF) can connect to any member node of the logical server.

SAP Sybase IQ ignores the connection parameter NodeType when you log in with the **SERVER** logical server context.

Password Connection Parameter [PWD]

Provides a password for the connection.

Usage
Anywhere

Default
No password provided

Description
Every database user has a password. The password must be supplied for the user to connect to the database. By default, the password has the same case-sensitivity as the data, and by default IQ databases are case sensitive.

The password parameter is not encrypted. If you are storing passwords in a connection profile, use the **EncryptedPassword** parameter. The SAP Sybase IQ ODBC configuration tool uses encrypted parameters.

If you specify both **Password** and **EncryptedPassword** are specified, **Password** takes precedence.

Examples

The following connection string fragment supplies the user ID `DBA` and password `SQL`.

```
uid=DBA;pwd=SQL
```

Alternatively, you can set these parameters in the **User ID** and **Password** text boxes in the connection window.

PrefetchBuffer Connection Parameter [PBUF]

Allows the user to set the maximum amount of memory.

Sets the maximum amount of memory, in bytes, for buffering rows.

Usage

Anywhere

Values

Integer { **k** | **m** }

Default

512 (KB)

Description

The **PrefetchBuffer** connection parameter controls the memory allocated on the client to store prefetched rows. The value is in bytes, but you can use k or m to specify units of kilobytes or megabytes. This connection parameter accepts values between 64KB and 8MB. In some circumstances, increasing the number of rows prefetched from the database server by the client can improve query performance. You can increase the number of rows prefetched using the **PrefetchRows** and **PrefetchBuffer** connection parameters.

Increasing the **PrefetchBuffer (PBUF)** connection parameter increases the amount of memory used to buffer GET DATA requests. This may improve performance for some applications that process many GET DATA (SQLGetData) requests.

Example

To determine if the PrefetchBuffer memory limit is reducing the number of prefetched rows, use this connection string fragment:

```
 ...prefetchrows=100;logfile=c:\ client.txt
```

To increase the memory limit to 256KB, use:

```
...prefetchrows=100;prefetchbuffer=256
```

PrefetchRows Connection Parameter [PROWS]

The PrefetchRows connection parameter sets the maximum number of rows to prefetch when querying the database.

Usage
Anywhere

Default
10 (200 when using the .NET Data Provider)

Description
Increasing the number of rows prefetched from the database server by the client can improve performance on cursors that do only fetch relative 0 or 1, with either single row or wide fetches. Wide fetches include embedded SQL array fetches and ODBC block fetches.

Improvements occur particularly under the following conditions:

- The application fetches many rows (several hundred or more) with very few absolute fetches.
- The application fetches rows at a high rate, and the client and server are on the same machine or connected by a fast network.
- Client/server communication is over a slow network, such as a dial-up link or wide area network.

The number of rows prefetched is limited both by the PrefetchRows connection parameter and the PrefetchBuffer parameter, which limits the memory available for storing prefetched rows.

The maximum number of rows that can be prefetched is 1000.

Example
The following connection string fragment sets the number of prefetched rows to 100:

```
...prefetchrows=100;...
```

RetryConnectionTimeout (RetryConnTO) connection parameter

Instructs the client library (DBLIB, ODBC, ADO, and so on) to keep retrying the connection attempt, as long as the server is not found, for the specified period of time.

Syntax

```
{ RetryConnectionTimeout | RetryConnTO }=timeout-value
```

Usage
Anywhere

Allowed values

- **timeout-value –** The value specified by this connection is a timeout, in seconds. It is not a counter of the number of times to retry the connection attempt. The default value of zero indicates that the connection attempt should only be tried once.

Default
0

Remarks
There is a half-second delay between iterations, and the retries only occur if the connection attempt failed because the database server was not found. Any other error is returned immediately. If the database server is not found, the connection attempt will take at least as long as the time specified by the RetryConnectionTimeout connection parameter.

The default TCP timeout is 5 seconds, so if your connection string contains a value for RetryConnTO that is less than 5, for example `Host=host-name;RetryConnTO=3`, then the connection attempt still takes 5 seconds.

Example

The following connection string fragment tells the client library to continue to retry the connection attempt for at least 5 seconds:

```
...RetryConnTO=5;...
```

ServerName Connection Parameter [ENG]

Synonym for EngineName connection parameter [ENG].

StartLine Connection Parameter [START]

Starts a database server running from an application.

Usage
Embedded databases

Default
No StartLine parameter

Description
Supply a StartLine parameter only if you are connecting to a database server that is not currently running. The StartLine parameter is a command line to start a server.

Example
The following data source fragment starts a database server with a cache of 32MB.

```
StartLine=dbeng6 -c 32M iqdemo.db
```

Unconditional Connection Parameter [UNC]

Stops a server using **dbstop** even when there are connections to the server.

Usage
Anywhere

Default
No

Description
The **dbstop** command line utility shuts down a database server. If you specify **Unconditional=Yes** in the connection string, the server is shut down even if there are active connections. If **Unconditional** is not set to **Yes**, the server shuts down only if there are no active connections.

Example
The following command line shuts down the server unconditionally:

```
dbstop -c "uid=DBA;pwd=SQL;eng=server-name;unc=yes"
```

See Also
Utility Guide > stop_iq Database Shutdown Utility

Userid Connection Parameter [UID]

Specifies the user ID with which you log on to the database.

Usage
Anywhere (DSN and FILEDSN connection parameters do not support verbose form of keyword)

Default
None

Description
You must always supply a user ID when connecting to a database. The user ID is not case-sensitive, and is unaffected by the setting of the Case Respect database property.

Examples
The following connection string fragment supplies the user ID `DBA` and password `SQL`:

```
uid=DBA;pwd=SQL
```

Network Communications Parameters

If you experience problems with client/server network communications, you can set a number of command line parameters for both the client and the server. These parameters enable you to work around peculiarities of different network protocol implementations.

Supply the network communication parameters on the server or client command line as in the following example:

```
start_iq -x tcpip(PARM1=value1;PARM2=value2;. . .),...
```

From the client side, communication parameters are entered as in this example:

```
CommLinks=tcpip(PARM1=value1;PARM2=value2;. . .),...
```

If they include spaces, the network communication parameters must be enclosed in quotation marks to be parsed properly:

```
start_iq -x "tcpip(PARM1=value 1;PARM2=value 2;...),..."
start_iq -x "tcpip(PARM1=value1;PARM2=value2;...)"
```

In UNIX, quotation marks are required if more than one parameter is given, because UNIX interprets the semicolon as a command separator.

Boolean parameters are turned on with any of YES, ON, TRUE, or 1, and are turned off with any of NO, OFF, FALSE, or 0. The parameters are case-insensitive.

Enter the commands to set the communication parameters on a single line; you can also include them in a configuration file, then use the @ server or client command line switch to invoke the configuration file.

TCP/IP, HTTP, and HTTPS Communications Parameters

The parameters currently available for TCP/IP, HTTP, and HTTPS are as follows.

TCP/IP	HTTP and HTTPS
Broadcast [BCAST]	Identity
BroadcastListener [BLISTENER]	Identity_Password
ClientPort [CPORT]	DatabaseName [DBN]
DLL	LocalOnly [LOCAL]
DoBroadcast [DOBROAD]	LogFile [LOG]
Host [IP]	LogMaxSize [LSize]
LocalOnly [LOCAL]	LogOptions [LOpt]
LDAP [LDAP]	LogFormat [LF]

TCP/IP	HTTP and HTTPS
MyIP [ME]	MaxConnections [MaxConn]
ReceiveBufferSize [RCVBUFSZ]	MaxRequestSize [MaxSize]
SendBufferSize [SNDBUFSZ]	MyIP [ME]
ServerPort [PORT]	ServerPort [PORT]
TDS	Timeout [TO]
Timeout [TO]	
VerifyServerName [VERIFY]	

Broadcast Communication Parameter [BCAST]

Specifies a broadcast address.

Usage
TCP/IP

Values
String (in the form of an IP address)

Default
Broadcasts to all addresses on the same subnet.

Description
The default broadcast address is created using the local IP address and subnet mask. The subnet mask indicates which portion of the IP address identifies the network, and which part identifies the host.

For a subnet of 10.24.98.x, with a mask of 255.255.255.0, the default broadcast address is 10.24.98.255.

When specifying an IPv6 address on a Windows platform, use the interface identifier. UNIX platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Example
To tell the client to broadcast only on interface number 2 when using IPv6, use:

```
LINKS=tcpip(BROADCAST=ff02::1%2)
```

BroadcastListener Communication Parameter [BLISTENER]

Controls broadcast listening for this port.

Usage
TCP/IP, server side

Values
YES, **NO**

Default
YES

Description
NO turns listening off for this port. Using **-sb 0** is the same as specifying `BroadcastListener=NO` on TCP/IP.

Example
To start a server that accepts TCP/IP connections that use BroadcastListener=NO:

```
start_iq -x tcpip(BroadcastListener=NO)
```

ClientPort Communication Parameter [CPort]

Designates the port number on which the client application communicates using TCP/IP. You may specify a single port number, or a combination of individual port numbers and ranges of port numbers.

Usage
TCP/IP (client side only)

Default
Assigned dynamically per connection by the networking implementation. If you do not have firewall restrictions, do not use this parameter.

Description
This option is provided for connections across firewalls, as firewall software filters according to TCP/UDP port. Do not use this parameter unless you need to for firewall reasons.

Specify a list or a range of port numbers to make multiple connections using a given data source or connect string. If you specify a single port number, your application can maintain only one connection at a time. In fact, even after closing the one connection, there is a short timeout period during which no new connection can be made using the specified port. When you specify a list or range of port numbers, the application keeps trying port numbers until it finds one to which it can successfully bind.

Examples

- The following string makes a connection from an application using port 6000 to a server named my_server using port 5000:

```
CommLinks=tcpip{ClientPort=6000;ServerPort=5000};
ServerName=my_server
```

- The following string makes a connection from an application that can use ports 5050 through 5060, as well as ports 5040 and 5070, for communicating with a server named my_server using the default server port:

```
CommLinks=tcpip{ClientPort=5040,5050-5060,5070};ServerName=my_serve
r
```

DatabaseName Communication Parameter [DBN]

Specifies the name of a database to use when processing web requests, or uses the REQUIRED or AUTO keyword to specify whether database names are required as part of the URI.

Usage
HTTP, HTTPS

Values
AUTO, **REQUIRED**, *database-name*

Default
AUTO

Description
If this parameter is set to REQUIRED, the URI must specify a database name.

If this parameter is set to AUTO, the URI may specify a database name, but does not need to do so. If the URI contains no database name, the default database on the server processes web requests. Since the server must guess whether or not the URI contains a database name when set to AUTO, you should design your web site so as to avoid ambiguity.

If this parameter is set to the name of a database, that database is used to process all web requests. The URI must not contain a database name.

Example
The following command starts two databases, but permits only one of them to be accessed via HTTP.

```
start_iq -xs http(dbn=web) iqdemo.db web.db
```

DoBroadcast Communication Parameter [DBROAD]

Performs a broadcast to search for servers.

Usage
TCP/IP (all platforms)

Values
ALL, **NONE**, **DIRECT** (client side)

YES, **NO** (server side)

Default
ALL

Description

- Client usage – with `DoBroadcast=ALL` (formerly `DoBroadcast=YES`) a broadcast is performed to search for a server. The broadcast goes first to the local subnet. If `HOST=` is specified, broadcast packets are also sent to each of the hosts. For TCP, all broadcast packets are UDP packets.
- With `DoBroadcast=DIRECT` (formerly `DoBroadcast=NO`), no broadcast is performed to the local subnet to search for a database server. Broadcast packets are sent only to the hosts listed in the HOST (IP) communication parameter. If you specify `DoBroadcast=DIRECT`, the HOST (IP) communication parameter is required.
- Specifying `DoBroadcast=NONE` causes no UDP broadcasts to be used. A TCP/IP connection is made directly with the HOST/PORT specified, and the server name is verified. With TCP/IP, you can choose not to verify the server name by setting the VerifyServerName (VERIFY) communication parameter to **NO**. The HOST (IP) communication parameter is a required parameter, while the ServerPort (PORT) communication parameter is optional.
- DIRECT and NONE, you must specify the server host with the HOST option.
- Server usage – setting `DoBroadcast=NO` prevents the database server from broadcasting to find other servers with the same name. This is useful in certain rare circumstances, but is not generally recommended.

Examples

- The following command starts a client without broadcasting to search for a database server. Instead, the server is looked for only on the computer named **silver**.
  ```
  dbisql -x tcpip(DOBROADCAST=DIRECT;HOST=silver) iqdemo
  ```
- On UNIX, the options must be enclosed in quotation marks:
  ```
  dbisql -x "tcpip(DOBROADCAST=DIRECT;HOST=silver)" iqdemo
  ```

Host Communication Parameter [IP]

Specifies additional machines outside the immediate network to be searched by the client library.

Usage

TCP/IP (all platforms) server and client sides

Description

On the server, the search is carried out to avoid starting a server with a duplicate name.

For TCP/IP, the *hostname* or a dot-separated IP address may be used.

The server prints this addressing information during startup if the **-z** switch is used. In addition, the application writes this information to its logfile if **LogFile** is specified (**Debug** is set to TRUE).

You can use a semicolon-separated list of addresses to search for more than one machine. Also, you can append a port number to an IP address using a colon as separator. Alternatively, you can specify the host and server ports explicitly, as in *Host=nnn.nn.nnn.nnn;ServerPort=pppp*.

IP and **HOST** are synonyms when using TCP/IP.

When specifying an IPv6 address on a Windows platform, use the interface identifier. UNIX platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Values

String

Default

No additional machines

Examples

- The following connection string fragment instructs the client to look on the machines “kangaroo” and 197.75.209.222 (port 2369) to find a database server called **iqdemo**:

  ```
  ...ENG=iqdemo CommLinks=tcpip(IP=kangaroo;IP=197.75.209.222:2369)
  ```

- For UNIX, quotation marks are required around the TCP/IP options:

  ```
  dbisql -x "tcpip(HOST=kangaroo;HOST=197.75.209.222)" iqdemo
  ```

- The following connection string fragment instructs the client to look on the machines my_server and kangaroo to find a database server. A connection is attempted to the first host that responds.

  ```
  dbisql -c
  "UID=DBA;PWD=sql;LINKS=tcpip(HOST=my_server,kangaroo;PORT=2639)"
  ```

Identity communication parameter

Specifies the name of an identity file.

Syntax
Identity=*identity-file*

Available protocols
HTTPS

Allowed values

- **identity-file –** This string specifies the name of an identity file.

Default
None

Remarks
When you use transport-layer security, the identity file contains the public certificate and its private key, and for certificates that are not self-signed, the identity file also contains all the signing certificates, which includes, among other things, the encryption certificate. The password for this certificate must be specified with the Identity_Password parameter.

Example
Start a database server that requires web connections to use a particular encryption certificate.

```
iqsrv16 -xs https(Identity=cert.file;Identity_Password=secret) ...
```

Identity_Password Communication Parameter

Specifies the password for the encryption certificate.

Syntax
Identity_Password=*password*

Available protocols
HTTPS

Allowed values

- **password –** This string specifies the password for an encryption certificate

Default
None

Remarks
When you use transport-layer security, this option specifies the password that matches the password for the encryption certificate specified by the Identity communication parameter.

Example
Start a database server that requires web connections to use a particular encryption certificate.

```
iqsrv16 -xs https(Identity=cert.file;Identity_Password=secret) ...
```

LDAP Communication Parameter [LDAP]

Allows clients running over a WAN or through a firewall to find servers without specifying the IP address.

Usage
Allows clients running over a WAN or through a firewall to find servers without specifying the IP address. It also allows the Locate utility (**dblocate**) to find such servers.

TCP/IP (server side only)

Values
YES, **NO**, or *filename*

Default
ON

The default *filename* is `asaldap.ini`

Description
Having the database server register itself with an LDAP server allows clients (and the Locate utility [**dblocate**]) to query the LDAP server.

Specifying **LDAP**=*filename* turns LDAP support on and uses the specified file as the configuration file. Specifying **LDAP**=*YES* turns LDAP support on and uses `saldap.ini` as the configuration file.

LDAP is only used with TCP/IP, and only on network servers.

LocalOnly Communication Parameter [LOCAL]

Allows a client to choose to connect only to a server on the local machine, if one exists.

Usage
TCP/IP, HTTP, HTTPS

Values
YES, **NO**

Default
NO

Description
If no server with the matching server name is found on the local machine, no server is autostarted.

The LocalOnly (LOCAL) communication parameter is useful only if `DoBroadcast=ALL` (the default)

`LocalOnly=YES` uses the regular broadcast mechanism, except that broadcast responses from servers on other machines are ignored.

You can use the LocalOnly (LOCAL) communication parameter with the server to restrict connections to the local machine. Connection attempts from remote machines do not find this server, and the Locate [**dblocate**] utility do not see this server. Running a server with the LocalOnly (LOCAL) communication parameter set to YES allows the network server to run as a personal server without experiencing connection or CPU limits.

LogFile Communication Parameter [LOG]

Specifies the name of the file to which the database server is to write information about web requests.

Usage
HTTP, HTTPS

Values
Filename

Default
None

LogFormat Communication Parameter [LF]

Controls the format of messages written to the log file and which fields appear in them.

Usage
HTTP, HTTPS

Values
Format-string

Default
@T - @W - @I - @P - "@M @U @V" - @R - @L - @E

Description

If the message string contains any of the following codes, the current values are substituted for the codes as each message is written.

- The @ character
- Date and time that processing of the request started, unless the request cannot be queued due to an error
- Date and time that the client connected
- Name of the database associated with the request
- Text of the error message, if an error occurred
- Date and time that processing of the request finished
- IP address of the client
- Length of the response, in bytes, including headers and body
- HTTP request method
- Listener port associated with the request
- Date and time that the request was queued for processing, unless the request could not be queued due to an error
- Status code and description of the HTTP response
- HTTP status code
- Date and time that the current log entry was written
- Requested URI
- Requested HTTP version
- Time taken to process the request (@F – @B), or 0.000 if the request was not processed due to an error

LogMaxSize (LSIZE) protocol option

Controls the maximum size of the log file where the database server writes information about web requests.

Syntax

```
{ LogMaxSize | LSIZE }=size[ k | m | g  ]
```

Available protocols

HTTP, HTTPS

Allowed values

- **size –** This integer specifies the maximum size of the file where web request information is written. The default value is in bytes, but you can use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes, respectively. If LogMaxSize is zero, the log file size is unlimited.

Default
0

Remarks
When the log file reaches the stated size, it is renamed and another log file is created.

LogOptions Communication Parameter [LOPT]

Controls which categories of messages are logged.

Usage
HTTP, HTTPS

Values
NONE, **OK**, **INFO**, **ERRORS**, **ALL**, *status-codes*, **REQHDRS**, **RESHDRS**, **HEADERS**

Default
ALL

Description
The values available include keywords that select particular types of messages, and HTTP status codes. Multiple values may be specified, separated by commas.

The following keywords control which categories of messages are logged:

- Log nothing.
- Log requests that complete successfully (20x HTTP status codes).
- Log requests that return over or not modified status codes (30x HTTP status codes).
- Log all errors (40x and 50x HTTP status codes)
- Log all requests.

The following common HTTP status codes are also available. They can be used to log requests that return particular status codes:

- OK
- Bad request
- Unauthorized
- Forbidden
- Not found
- Request timeout
- Not implemented
- Service unavailable

In addition, you can use these keywords to obtain more information about the logged messages:

- When logging requests, also write request headers to the log file.
- When logging requests, also write response headers to the log file.
- When logging requests, also write both request and response headers to the log file (same as REQHDRS,RESHDRS).

MaxConnections Communication Parameter [MAXCONN]

Specifies the number of simultaneous connections accepted by the server.

Usage
HTTP, HTTPS

Values
Number

Default
Number of licensed connections

Description
The value 0 indicates no limit.

MaxRequestSize (MAXSIZE) protocol option

Specifies the size of the largest request the database server can accept.

Syntax

```
{ MaxRequestSize | MAXSIZE }=size[ k | m | g  ]
```

Available protocols
HTTP, HTTPS

Allowed values

- **size –** This integer specifies the size of the largest request the database server can accept. The default value is in bytes, but you can use `k`, `m`, or `g` to specify units of kilobytes, megabytes, or gigabytes, respectively. The value 0 disables this limit, but should be used with extreme caution. Without this limit, a rogue client could overload the database server or cause it to run out of memory.

Default
100k

Remarks
If the size of a request exceeds this limit, the connection is closed and the client returns a response indicating that the request is too large. This value limits only the size of the request, not that of the response.

Example

The following command instructs the database server to accept requests up to 150000 bytes in size:

```
iqsrv16 -xs http(MaxRequestSize=150000)
```

MyIP Communication Parameter [ME]

Controls whether the client attempts to determine addressing information.

Usage
TCP/IP, HTTP, HTTPS

Values
String

Description
The MyIP (ME) parameter is provided for computers with more than one network adapter.

Each adapter has an IP address. By default, SAP Sybase IQ uses the first network card it finds. If you want your database server to use more than one network card, specify the address of each card in the MyIP (ME) parameter.

If you supply the keyword NONE as the IP number, no attempt is made to determine the addressing information. NONE is intended primarily for clients on operating systems where this operation is expensive, such as machines that have multiple network cards or remote access (RAS) software and a network card. It is not intended for use on the server.

On Windows platforms, you can use this option multiple times for machines with multiple IP addresses.

Separate multiple IP addresses with commas. You can optionally append a port number to the IP address, separated by a colon.

When specifying an IPv6 address on a Windows platform, use the interface identifier. UNIX platforms support both interface identifiers and interface names in IPv6 addresses. The interface identifier is required on Linux (kernel 2.6.13 and later).

Examples

- This Windows command line (entered all on one line) instructs the server to use two network cards, one with a specified port number:

```
start_iq -x tcpip(MyIP=192.75.209.12:2367,192.75.209.32)
c:\sybase\IQ-16_0\demo\iqdemo.db
```

- This command line (entered all on one line) instructs the server to use an IPv6 network card:

```
start_iq -x tcpip(MyIP=fe80::5445:5245:444f)
"c:\sybase\IQ-16_0\demo\iqdemo.db"
```

- This connection string fragment instructs the client to make no attempt to determine addressing information:

```
...CommLinks= tcpip(MyIP=NONE)...
```

PreFetchOnOpen Communication Parameter

Sends a prefetch request with a cursor open request, thereby eliminating a network request to fetch rows each time a cursor is opened.

Usage
ODBC

Values
YES, **NO**

Default
NO

Description
Columns must already be bound for the prefetch to occur on the open. Rebinding columns between the cursor open and the first fetch when using PreFetchOnOpen reduces performance.

Calling ODBC's SQLExecute or SQLExecDirect on a query or stored procedure which returns a result set causes a cursor open.

Enabling this option can improve performance if your:

- Network exhibits poor latency
- Application sends many cursor open and close requests

ReceiveBufferSize Communication Parameter [RCVBUFSZ]

Sets the size for a buffer used by the TCP/IP protocol stack. You may want to increase the value if LOB performance over the network is important.

Usage
TCP/IP

Values
Integer [**k** | **m** | **g**]

v

Maximum Allowed Value
1048576 bytes (1MB)

Default
Machine-dependent

Description
Use **k**, **m**, or **g** to specify units of kilobytes, megabytes, or gigabytes, respectively.

SendBufferSize Communication Parameter [SNDBUFSZ]

Sets the size for a buffer used by the TCP/IP protocol stack.

Usage
TCP/IP

Values
Integer [**k** | **m** | **g**]

Maximum Allowed Value
1048576 bytes (1MB)

Default
Computer-dependent

Description
The default value is in bytes, but you can use **k**, **m**, or **g** to specify kilobytes, megabytes, or gigabytes, respectively. You may want to increase the value if LOB performance over the network is important.

ServerPort Communication Parameter [PORT]

In the case of the database server, the **ServerPort** option designates the port number on which to communicate using TCP/IP. In a data source, the **ServerPort** option informs the client of the port or ports on which database servers are listening for TCP/IP communication.

Usage
TCP/IP (all platforms), HTTP, HTTPS

Values
Integer

Default
The default value for TCP/IP is **2638.** The default value for HTTP is 80. The default value for HTTPS is 443.

Description

The Internet Assigned Numbers Authority has assigned the IQ database server port number 2638 to use for TCP/IP communications. However, applications are not disallowed from using this reserved port, and this may result in an addressing collision between the database server and another application.

The client broadcasts to every port that is specified on the **ServerPort** parameter to find the server. If ServerPort is unspecified, the default port 2638 is used.

The database server uses the port number specified in **ServerPort** for TCP/IP connections. If ServerPort is unspecified, the default port 2638 is used. Hence, applications can connect to the database server without specifying a port number. An exception is the HP-UX operating system, on which the server does not listen on port 2638 if it is started on another port.

By default, the database server listens on the standard HTTP and HTTPS ports of 80 and 443, respectively.

Example

1. On Windows, start an IQ network server:

   ```
   start_iq -x tcpip c:\sybase\IQ-16_0\demo\iqdemo.db
   ```

   Port number 2638 is now taken.
2. Attempt to start another database server:

   ```
   start_iq -x tcpip c:\sybase\IQ-16_0\demo\iqdemo2.db
   ```

   The default port being allocated, the server starts on another port.
3. Start another database server on port 2639:

   ```
   start_iq -x "tcpip(ServerPort=2639)" c:\sybase\IQ-16_0\demo
   \iqdemo3.db
   ```

   This succeeds as long as 2639 is not a reserved port and no other application has allocated it.
4. If another web server on your machine is already using port 80 or you do not have permission to start a server on such a low port number, start a server that listens on an alternate port, such as 8080:

   ```
   start_iq -xs http{port=8080} -n server3 web.db
   ```

Sessions Communication Parameter

Sets the maximum number of clients that can communicate with the server at one time through a single LAN adapter.

Usage

NetBIOS. Server side only.

Description
The default setting is operating-system specific. The value is an integer, with maximum value 254.

NetBIOS network software has a limit to the number of commands allowed per machine. SAP Sybase IQ uses these NetBIOS commands, and disallows further connections if the system has no more commands available, even if the number of NetBIOS commands is fewer than the value of the Sessions parameter.

Default
OS-specific. On Windows, the default is 16.

Example
The following statement starts a server with a database named `iqdemo`, allowing 200 NetBIOS connections:

```
start_iq -x netbios(sessions=200) iqdemo.db
```

TDS Communication Parameter

Controls whether the server allows TDS connections.

Usage
TCP/IP, NamedPipes (server side only)

Values
YES, **NO**

Default
YES

Description
To disallow TDS connections to a database server, set TDS to NO. To ensure that only encrypted connections are made to your server, these port options are the only way to disallow TDS connections.

Example
The following command starts a database server that uses the TCP/IP protocol, but disallows connections from Open Client or jConnect applications.

```
start_iq -x tcpip(TDS=NO) ...
```

Timeout Communication Parameter [TO]

Specifies the length of time, in seconds, to wait for a response when establishing communications and when disconnecting.

Usage
TCP/IP (all platforms), HTTP, HTTPS

Values
Integer, in seconds

Maximum Allowed Value
3600 seconds

Default
5

Description
You may want to try longer times if you are having trouble establishing TCP/IP communications.

In HTTP or HTTPS applications, this parameter specifies the maximum idle time permitted when receiving a request. If this limit is reached, the connection is closed and a `408 REQUEST TIMEOUT` is returned to the client. The value 0 disables idle timeout, and should be used with extreme caution. Without this limit, a rogue client might consume the server's resources and prevent other clients from connecting.

Example
The following data source fragment starts a TCP/IP communications link only, with a timeout period of twenty seconds.

```
...
CommLinks=tcpip(TO=20)
...
```

VerifyServerName Communication Parameter [Verify]

The VerifyServerName Communication Parameter specifies whether the server name is verified when connecting to this host.

Usage
TCP/IP (Client side only)

Values
YES, **NO**

Default
YES

Description
Normally, you should not set this option. It is used only for connecting to multiplex secondary servers to balance query loads among these servers.

When connecting over TCP using the `DoBroadcast=NONE` parameter, the client makes a TCP connection, then verifies that the name of the server found is the same as the one it is looking for. Specifying `VerifyServerName=NO` skips the verification of the server name. This allows IQ clients to connect to a SAP Sybase IQ server if they know only an IP address/port.

The server name must still be specified in the connection string, but it is ignored. The VerifyServerName (VERIFY) communication parameter is used only if `DoBroadcast=NONE` is specified.

When used as shown in the example, setting this option to NO lets you specify a connection to a particular IP address and port number. The IP address and port number are for a load balancing machine that acts as a gateway between the IQ client and the IQ server.

Example
To use this option, on the client machine, create a new ODBC DSN in the ODBC Administrator, and specify parameters as follows:

- On the Database tab, specify a generic server name for connecting to all of the secondary servers, for example, `qserv`. A server name is required, but ignored because of parameters in the Network tab.
- On the Network tab, select the TCP/IP check box and type in the text box:

  ```
  host=hostname;port=port#;DOBROADCAST=NONE;VERIFY=NO
  ```

  For example:

  ```
  host=hostname;port=2222;DOBROADCAST=NONE;VERIFY=NO
  ```

When an IQ client connects to this DSN, the load balancer dispatches the connection to a particular secondary server based on the workload of the machine.

Appendix: SQL Statements and Options Reference

Certain SQL Statements and database options have special syntax to support SAP Sybase IQ database administration.

Database Options

Database options modify database behavior.

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.

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.

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.

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
```

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 = ''
```

SQL Statements

SQL statements perform data definition and manipulation tasks.

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).

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 dbfile 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:

```
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 dbfile 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 dbfile 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.

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.

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 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_COORDINA-TOR_AS_MEMBER	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. • **Values –** ON, OFF • **Default –** ON
DQP_ENABLED	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. • **Values –** 0, 1, 2 • **Default –** 1
LOGIN_REDIRECTION	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. • **Values –** ON, OFF • **Default –** OFF

Option	Description
REDIRECTION_WAIT-ERS_THRESHOLD	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. • **Values –** Integer • **Default –** 5
TEMP_DA-TA_IN_SHARED_TEMP	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. • **Values –** ON, OFF • **Default –** OFF

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 dbsp1,
     P2 VALUES <= (200) IN dbsp2,
     P3 VALUES <= (MAX) IN dbsp3
    ) IN dbsp4);
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,
       c2 DATE,
       c3 VARCHAR(10))
   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);
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 `dbsp5`:

```
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 dbsp4,
    P42 VALUES <= ('2008-12-31') IN dbsp4);
```

  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 dbsp3,
   P32 VALUES <= ('2007-12-31') IN dbsp3);
```

  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 dbsp2,
    P22 VALUES <= ('2006-12-01') IN dbsp2);
```

  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:

- 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	Requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER privilege on the underlying table • You own the underlying table UNIQUE, PRIMARY KEY, FOREIGN KEY, or IQ UNIQUE column constraint – Requires above along with REFERENCE privilege on the underlying table. FOREIGN KEY table constraint requires above along with one of: • CREATE ANY INDEX system privilege • CREATE ANY OBJECT system privilege • REFERENCE privilege on the base table PARTITION BY RANGE requires above along with one of: • CREATE ANY OBJECT system privilege • CREATE permission on the dbspaces where the partitions are being created
Alter	Requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table • You own the table. To alter a primary key or unique constraint, also requires REFERENCE permission on the table.

Clause	Privilege Required
Drop	Drop a column with no constraints – Requires one of: • ALTER ANY OBJECT system privilege • ALTER ANY TABLE system privilege • ALTER permission on the underlying table • You own the underlying table Drop a column or table with a constraint requires above along with REFERENCE permission if using ALTER permission. Drop a partition on table owned by self – None required. Drop a partition on table owned by other users – Requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table
RENAME	Requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table • You own the table

Clause	Privilege Required
Move	Requires one of: • 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 Also requires one of the following: • CREATE ANY OBJECT system privilege • CREATE privilege on the dbspace to which the partition is being moved
Split Partition	Partition on table owned by self – None required. Partition on table owned by other users – Requires one of: • SELECT ANY TABLE system privilege • SELECT privilege on table Also requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER privilege on the table
Merge Partition, Unpartition	Table owned by self – None required. Table owned by other users – Requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER privilege on the table

Clause	Privilege Required
Partition By	Requires one of: • CREATE ANY OBJECT system privilege • CREATE permission on the dbspaces where the partitions are being created Also requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege • ALTER permission on the table • You own the table
Enable or disable RLV store	Requires one of: • ALTER ANY TABLE system privilege • ALTER ANY OBJECT system privilege

See also

- *Restrictions* on page 163
- *Range Partitions* on page 165
- *Hash Partitions* on page 166
- *Hash-Range Partitions* on page 167

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.

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.

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 }

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 29. Collation Tailoring Option for SAP Sybase IQ

Keyword	Collation	Alternate Forms	Allowed Values
CaseSensitivity	All supported collations	CaseSensitive, Case	• **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 30. 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/rdsk/bigdb'
MESSAGE PATH '/s1/data/bigdb.iqmsg'
TEMPORARY PATH '/dev/md/rdsk/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.

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.

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 31. 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:

```
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
  - 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.

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.

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 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.

LS Policy Options

Available options for root and user-defined LS policies.

Option	Description
ALLOW_COORDINATOR_AS_MEMBER	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. • **Values –** ON, OFF • **Default –** ON
DQP_ENABLED	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. • **Values –** 0, 1, 2 • **Default –** 1
LOGIN_REDIRECTION	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. • **Values –** ON, OFF • **Default –** OFF

Option	Description
REDIRECTION_WAIT-ERS_THRESHOLD	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. • **Values –** Integer • **Default –** 5
TEMP_DA-TA_IN_SHARED_TEMP	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. • **Values –** ON, OFF • **Default –** OFF

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 )
   | 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*(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(col1 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	Table owned by self – Requires CREATE privilege on the dbspace where the table is created. Also requires one of: • CREATE TABLE system privilege. • CREATE ANY OBJECT system privilege. Table owned by any user – Requires CREATE privilege on the dbspace where the table is created. Also requires one of: • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege.
Global temporary table	Table owned by self – Requires one of: • CREATE TABLE system privilege. • CREATE ANY OBJECT system privilege. Table owned by any user – Requires one of: • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege.
Proxy table	Table owned by self – Requires one of: • CREATE PROXY TABLE system privilege. • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege. Table owned by any user – Requires one of: • CREATE ANY TABLE system privilege. • CREATE ANY OBJECT system privilege.

See also

- *Restrictions* on page 163
- *Range Partitions* on page 165
- *Hash Partitions* on page 166
- *Hash-Range Partitions* on page 167

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 TABLEis 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:

- 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 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.

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.

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: `monitor_options -summary`
-cache	Displays main or temp buffer cache activity in detail. Critical fields are `Finds`, `HR%`, and `BWaits`. Usage: `monitor_options -cache`
-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: `monitor_options -cache_by_type`
-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: `monitor_options -file_suffix {extension}`
-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: `monitor_options -io`

Option	Description
-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: `monitor_options -bufalloc`
-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: `monitor_options -contention`
-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: `monitor_options -threads`
-interval	Specifies the reporting interval in seconds. The default is every 60 seconds. The minimum is every 2 seconds. 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: `monitor_options -interval`
-append | -truncate	Append or truncate output to existing output file. Truncate is the default. Usage: `monitor_options -append | -truncate`
-debug	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: `monitor_options -debug`

- **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

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:

```
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

.

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.

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

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.

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.

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

Index

-c switch 17
-contention 246
-iqnotemp 343

A

Access
    ODBC configuration for 83
adding
    TEXT index 182
AES encryption algorithm
    CREATE DATABASE statement 443
aggregates 181
ALLOW_SNAPSHOT_VERSIONING 234
ALTER DBSPACE statement
    ADD parameter 131
    examples 131
    SIZE parameter 132
    syntax 416
ALTER INDEX statement
    errors 420
ALTER LOGICAL SERVER statement
    syntax 422
ALTER LS POLICY statement
    syntax 424
ALTER statement
    automatic commit 218
ALTER TABLE statement
    foreign keys 209
    syntax 426
ALTER VIEW statement
    RECOMPILE 426
analyzing output 354
AppInfo connection parameter 364
AStart connection parameter 366
AStop connection parameter 367
AUDITING option 315
AUTOINCREMENT
    default 202
    negative numbers 202
AUTOINCREMENT column default 467
AutoPreCommit connection parameter 365
AutoStart connection parameter 366
AutoStop connection parameter 367
AVG function 181

B

backslashes
    Windows raw devices 112
backups
    concurrency 252
base tables 157
batches
    about 255
BCAST communication parameter
    description 393
BEGIN TRANSACTION statement
    remote data access 254
BIT data
    indexes allowed in 188
BLISTENER communication parameter
    description 394
block size 115
blocked write access
    determining blocking writers 337
    managing contention 337
blocking 226
BLOCKING option 224
BLOCKING_TIMEOUT option 225
Broadcast communication parameter
    description 393
BroadcastListener communication parameter
    description 394
buffer cache
    insufficient space 322, 335
    IQ UTILITIES command 324
    monitor 324
    partitioning 12, 13
buffer space
    connection parameter 370
bugs
    reporting 356

C

cache size
    setting for catalog store 17, 18
CACHE_PARTITIONS option 13

CALL statement
examples 256
case sensitivity
command line 6
connection parameters 363
catalog files
growth 330
catalog store
about 121
preallocating space 121
setting cache size 17, 18
validating indexes 215
CBSize connection parameter 369
CBSpace connection parameter 370
character sets
connection parameter 367
CharSet connection parameter 367
CHECK conditions
about 467
columns 205
deleting 207
modifying 207
tables 206
user-defined data types 206
CHECK ON COMMIT clause
referential integrity 467
checklist for Technical Support 361
CHECKPOINT statement
syntax 440
checkpoints
about 248
adjusting interval 339
automatic and explicit 248
in system recovery 251
choosing drivers
using the iAnywhere JDBC driver 58
using the jConnect JDBC driver 58
CLASSPATH environment variable
jConnect 59
client process information 364
client side
Encryption [ENC] connection parameter 378
ClientPort communication parameter 394
CMP index 172
recommended use 172
restrictions 173
columns
adding 160
altering 426
changing 160
constraints 467
defaults 197
deleting 161
command-line switches 6
displaying 6
required 8
CommBufferSize connection parameter 369
CommBufferSpace connection parameter 370
COMMIT statement
remote data access 254
syntax 441
COMMIT TRANSACTION statement
Transact-SQL 441
CommLinks connection parameter 370
communication parameters
about 363
Broadcast [BCAST] 393
BroadcastListener 394
ClientPort 394
DatabaseName [DBN] 395
DOBROADCAST 396
HOST 397
LDAP [LDAP] 399
LocalOnly [LOCAL] 399
LogOptions [LOPT] 402
MaxConnections 403
MYIP 404
PreFetchOnOpen 405
ReceiveBufferSize [RCVBUFSZ] 405
SendBufferSize [SNDBUFSZ] 406
ServerPort 406
SESSIONS 407
TDS 408
TIMEOUT 409
communications
Encryption [ENC] connection parameter 378
parameters 392
troubleshooting 339
Compare index
See CMP index
concurrency
backups 252
data definition 235
insertions, deletions, and queries 235
locking tables 492
configuration files
using 7

configuration parameters
overriding 322
configuring
ODBC data sources 82
connecting
BroadcastListener [BLISTENER] communication parameter 394
firewalls 394
jConnect 61
LDAP communication parameter 399
connection handle 350
connection information
IQ message file 359
request log 359
srvlog file 359
connection parameter 385
connection parameters
about 363
case insensitivity 363
CharSet 367
ConnectionPool 372
data sources 79
default 51
Encryption [ENC] 378
Escape 379
IDLE 380
in connection strings 54
LANG 382
LazyClose [LCLOSE] 382
priority 364
connection pooling 100
connection strings
representing 55
ConnectionName connection parameter 372
ConnectionPool connection parameter 372
connections
DEDICATED_TASK option 412
embedded database 46
establishing 54
examples 42
Interactive SQL 43
jConnect URL 60
JDBC 52
JDBC SAP Sybase Open Client 52
learning roadmap 41
limiting concurrent 14
local database 43
logging 106, 412
overview 53
remote 58, 254
temporary 102
using data source 50
connectivity
iAnywhere JDBC driver 58
jConnect 58
constant expression defaults 203
constraints
effect on performance 196
Containment index
See WD index
COUNT DISTINCT
impact on index choice 185
COUNT function 181
CREATE DATABASE statement 110
IQ RESERVE parameter 131
IQ SIZE parameter 131
raw devices 109
syntax 443
TEMPORARY RESERVE parameter 131
TEMPORARY SIZE parameter 131
CREATE DBSPACE statement 140
RESERVE parameter 131
SIZE parameter 131
CREATE DOMAIN statement
syntax 453
CREATE FUNCTION statement
syntax 263
CREATE INDEX statement
syntax 455
table use 459
CREATE LOGICAL SERVER statement 463
CREATE LS POLICY statement
syntax 465
CREATE statement
automatic commit 218
concurrency rules 235
CREATE TABLE statement
example 156
syntax 467
CREATE TEXT INDEX 182
creating
column defaults 198
data types 453
TEXT index 182
CS connection parameter 367
current date and time defaults 201
cursors
database options 411

hold 253
in transactions 252
message logging 254
ODBC configuration 83
positioned operations 254
procedures 258
scrolling 253
sensitivity 253

D

data
duplicated 195
invalid 195
data definition language
about 151
concurrency rules 235
data definitions
creating 155
data integrity
column defaults 197
constraints 197
overview 195
rules in the system tables 215
data link layer
troubleshooting 341
data source name
ODBC 83
data sources
about 79
command line creation 81
configuring 82
connecting with 50
ODBC 79
UNIX 84
using with jConnect 58
data types
creating 453
dropping user-defined 484
specifying in table creation 157
database
naming conflict 319
repair 317
database files
altering 416
database options
cursors 411
DEDICATED_TASK 412
duration 411
maximum string length 496
database server
as Windows service 6
command-line switches 6
connecting 58
name switch 9
naming at startup 10
remote 58
starting 3, 4
starting at command prompt 5
starting from Start menu 6
starting on Windows 6
stopping 35, 38
database servers
preventing from starting 366
stopping 391, 499
database utilities 51
database utility
log translation 311
DatabaseFile connection parameter 373
DatabaseName communication parameter
description 395
DatabaseName connection parameter 374
databases
block size 115
choosing a location 111
connecting to 53, 58
creating 107, 443
creating objects 155
default characteristics 110
designing 151
dropping 116
embedded 363, 367
file location 109
initial size 125
initializing 107
moving 108
naming 9
naming at startup 9
overview of setup 1
page size 113
permission to start 40
preallocating space 119
privileges 21
privileges needed to create 2
relative path names 111
schema creation 155
size 113
SQL Anywhere 121
starting 39

stopping 40, 498
unloading 39
URL 60
working with objects 151
DatabaseSwitches connection parameter 375
DataSourceName connection parameter 375
DATE index 173
additional indexes 177
advantages 176
comparison to other indexes 177
disadvantages 176
recommended use 173
restrictions 176
DATEPART
queries 173
Datetime index
See DTTM index
DBF connection parameter 373
embedded databases 46
dbisql
command line parameters 44
connection parameter examples 42
connections 44
logon window 45
port number 44
troubleshooting 328
DBKEY connection parameter 376
DBN communication parameter
description 395
DBN connection parameter 374
dbo user ID
views owned by 153, 484
DBS connection parameter 375
dbspace
adding on raw devices 109
changing the size 131
create example 131
creating 140
definition 119
displaying index information 134
displaying usage information 134
dropping 141
file location 109
file location when creating 108
IQ_SHARED_TEMP 119, 122
IQ_SYSTEM_MSG 124
IQ_SYSTEM_TEMP 119
management example 131
naming 140
offline 119
out-of-dbspace condition 115
read-only 143
SYSTEM 119
DBSpaceLogger event 332
dbspaces
altering 416
created automatically 108
dropping 484
monitoring space usage 332
out of space error messages 328
out-of-dbspace condition 328
setting offline 416
dbtran utility 311
exit codes 315
syntax 311
DDL 244
about 151
DDL locking 235
deadlock 226
detecting 324
resolving 324
deadlocks
reporting 227
DEDICATED_TASK option
description 412
default configuration file 7
default index
Flat FP 177
NBit 177
Tokenized FP (NBit) 177
defaults
AUTOINCREMENT 202
column 197
connection parameters 51
constant expressions 203
creating 198
current date and time 201
inserting 199
loading 199
NEWID 203
NULL 203
string and number 203
USER special value 201
deleting
column defaults 199
delimiters
example 455

Delphi
ODBC configuration for 83
device types
for databases 2
diagnostic tools 343
checking database options 351
checking server startup options 351
communications issues 356
logging server requests 352
sa_server_option 352
sp_iqcheckdb 351
sp_iqcheckoptions 351
sp_iqconnection 351
sp_iqcontext 351
sp_iqdbstatistics 351
sp_iqstatus 343
DisableMultiRowFetch connection parameter 376
disconnecting 105
disk
monitoring space usage 331
out of space 322, 323, 328
disk space
allocating 140
indexes 188
saving 253
distributed query processing
performance 424, 465
DMRF connection parameter 376
DOBROADCAST communication parameter 396
domains 453
DQP
performance 424, 465
driver
JDBC 52
drivers
jConnect JDBC driver 58
SQL Anywhere JDBC driver 58
DROP DATATYPE statement
syntax 484
DROP DBSPACE statement
syntax 484
DROP DOMAIN statement
syntax 484
DROP EVENT
syntax 484
DROP FUNCTION statement
syntax 484
DROP INDEX statement
syntax 484
DROP LOGICAL SERVER statement 487
DROP MESSAGE
syntax 484
DROP PROCEDURE statement
syntax 484
DROP statement
automatic commit 218
concurrency rules 235
syntax 484
DROP TABLE
IDENTITY_INSERT option 484
DROP TABLE statement
example 161
syntax 484
DROP VIEW statement
example 155
restriction 153, 484
syntax 484
dropping
views 153, 484
dropping partitions 426
DSN connection parameter 375
about 79
DTTM index 173
additional indexes 177
advantages 176
comparison to other indexes 177
disadvantages 176
recommended use 173
DumpAllThreads file 324

E

embedded databases 363, 367
connecting 46
Java 46
starting 46
ENC connection parameter
description 378
EncryptedPassword connection parameter 377
encryption
communications 408
Encryption [ENC] connection parameter 378
strong 378
encryption algorithms
CREATE DATABASE statement 443
Encryption connection parameter
description 378
ENG connection parameter 377
EngineName connection parameter 377

ENP connection parameter 377
entity integrity
    enforcing 207
environment variables
    SQLCONNECT 51
errors
    errors 235
    out-of-dbspace condition 328
    transaction processing 235
Escape connection parameter 379
Ethernet 342
events
    DBSpaceLogger 332
    dropping 484
    monitoring disk space usage 331
    monitoring space usage 332
    retrieving a schedule name 309
    retrieving an event name 309

F

fast projection indexes 177
fetch operation
    suppressing warnings 83
file size
    controlling 330
FileDataSourceName connection parameter 380
FileDSN
    connection parameter 79
    creating 85
    distributing 86
files
    dbspaces 416
    setting offline 416
    setting online 416
firewalls
    BroadcastListener [BLISTENER]
        communication parameter 394
    connecting across 394
    LDAP communication parameter 399
Flat FP 177
foreign key constraints
    removing 193
foreign keys
    existing unenforced 211
    integrity constraints 467
    optional 211
    unnamed 467
frame type 342
functions
    creating 263
    dropping 484
    user-defined 263

G

getiqinfo script 357
global temporary tables 245
    about 157
gm switch 14
    effect on recovery 251
GROUP BY clause
    impact on index choice 185
gt switch 17

H

hash partitions
    examples 166
    hash-partition-key 166
    hash-partitioning-scheme 166
    restrictions 166
HASH_THRASHING_PERCENT option 336
hash-range partitions
    examples 167
    hash-range-partitioning-scheme 167
    SUBPARTITION BY RANGE 167
HG index
    additional indexes 179
    advantages 179
    automatic creation 179
    comparison to other indexes 179
    disadvantages 179
    foreign key constraint 178
    multicolumn with NULL 180, 461
    NULL values 180, 461
    query performance 180
    recommended use 178
HG indexes
    retaining when removing foreign key
        constraints 193
High_Group index
    See HG index
High_Non_Group index
    See HNG index
HNG index 180
    additional indexes 181
    advantages 181
    comparison to other indexes 181
    disadvantages 181

recommended use 180
hold cursors 253
HOST communication parameter 397
hyperthreading
server switch 19

I

IANA
port number 406
iAnywhere JDBC driver
choosing a JDBC driver 61
IDENTITY column
and DROP TABLE 484
IDENTITY_INSERT option
dropping tables 484
IDLE connection parameter 380
import
jConnect 60
index advisor 192
index types
criteria for choosing 184
LF 181
recommendations 186
recommended 170
selecting 187
indexes
about 169
adding after loading tables 191
advice in message log 192
created automatically 159
creating 155, 190, 255, 455
default 177
disk space usage 188
dropping 193, 484
fast projection 177
Flat FP 177
in system views 193
listing 192
maximum unique values 336
multicolumn 460
multicolumn HG and NULL 180, 461
naming 459
NBit 177
owner 455
parallel creation 191
rebuilding 160
recommended combinations 190
renaming 192
selecting an index type 187
showing information about 192
table use 459
TEXT indexes 182
Tokenized FP (NBit) 177
too many on table 335
unique 455
INSERT statement 191
and integrity 196
inserting
column defaults 199
insufficient buffers
buffer cache 335
insufficient space 330
buffer cache 335
INT connection parameter 381
INTEGRATED LOGIN statement
grant 488
integrated logins
default user 100
network aspects 99
operating systems 95
using 97
integrity
column defaults 197
constraints 196, 197
overview 195
interface libraries
connections 53
IP address 364
ping 341
IP communication parameter 397
IPv6 addresses 404
IPX
server configuration 392
IQ PAGE SIZE 113
IQ store
raw device access 127
raw device permissions 128
IQ UNIQUE
changing value of 160
IQ UNIQUE constraint 204
IQ UNIQUE table option 159
IQ UTILITIES
buffer cache monitor 324
IQ UTILITIES statement
syntax 488
IQ_SHARED_TEMP 123
IQ_SYSTEM_MAIN
size guidelines 125

IQ_SYSTEM_MSG dbspace 124
IQ_SYSTEM_TEMP 343
iqdsn command 81
iqgovern switch 15
iqmc switch 11
iqmt switch 16
iqnumbercpus
    server switch 19
iqpartition startup switch 12
iqpartition switch 11
iqwmem switch 16
isolation level
    ODBC configuration 83
isolation levels 221

J

Java
    memory requirements 17
    use in SAP Sybase IQ 52
jConnect
    about 59
    choosing a JDBC driver 61
    CLASSPATH environment variable 59
    installation 59
    jdbcdrv.zip 59
    packages 60
    URL 60
    versions 59
jConnect driver
    connecting to IQ databases 58
JDBC
    connecting to a database 60
    jConnect 59
JDBC connectivity
    about 58
JDBC drivers
    choosing 61
    compatibility 61
    performance 61
joins
    performance 194
    performance impact 194

L

LANalyzer 342
LANG connection parameter
    description 382
LazyClose connection parameter
    description 382
LCLOSE connection parameter
    description 382
LDAP communication parameter
    description 399
LDAP servers
    LDAP communication parameter 399
LF communication parameter
    description 400
LF index 181
    additional indexes 182
    advantages 182
    comparison to other indexes 182
    disadvantages 182
    exceeding maximum unique values 336
    recommended use 181
links
    symbolic 443
Links connection parameter 370
LivenessTimeout connection parameter 383
load performance
    iqpartition server option 12
    table lock contention 246
LOAD TABLE statement 191
loading data
    column defaults 199
    concurrency rules 235
    errors 336
    monitoring space usage 331
    notification messages 346
    performance 335
loading schema
    recommended database size 125
loads
    scalability 13
LOCAL communication parameter
    description 399
local temporary tables 245
    about 157
local write-intent 244
LocalOnly communication parameter
    description 399
lock contention
    iqpartition server option 12
LOCK TABLE
    syntax 492
locking
    DDL operations 235

tables 234, 492
locks
managing contention 337
releasing with ROLLBACK 494
LOG communication parameter
description 400
log files
correlating connection information 359
server 33
LOG_CONNECT database option 412
LogFile
communication parameter 400
connection parameter 384
LogFormat communication parameter
description 400
logging connections 106
logical server 104
in simplex 104
logical server policies
altering 424
creating 465
defining 463
logical server policy
options 425, 466
logical servers 103
altering 422
LogicalServer
connection parameter 384
login redirection 424
LOGIN_MODE database option
integrated logins 96
logins
integrated 95, 96
See also connections
LogOptions communication parameter
description 402
LOPT communication parameter
description 402
Low_Fast index
See LF index
LS policy 425, 466
LTO connection parameter 383

M

main store
space management 115
managing
transactions 254
materialized views
dropping 484
MAX_TEMP_SPACE_PER_CONNECTION option 115
MAXCONN communication parameter
description 403
MaxConnections communication parameter
description 403
MDSR encryption algorithm
CREATE DATABASE statement 443
memory
creating wired memory pool 16
for catalog store cache 17
memory message
load notification messages 347
message file
connection information 359
message log
index recommendations 192
IQ_SYSTEM_MSG dbspace 124
messages
dropping 484
memory notification 347
out-of-dbspace condition 328
metadata
in catalog store 121
Microsoft Access 335
ODBC configuration for 83
Microsoft Visual Basic
ODBC configuration for 83
MIN_PASSWORD_LENGTH option 413
mklink utility 130
modifying
column defaults 199
modifying and deleting column defaults 199
monitor
in IQ UTILITIES statement 488
starting and stopping 488
monitor output options
-contention 246
multicolumn indexes 455, 460
multiplex
shared temporary storage 123
multiplex databases
creating 443
multiplex servers
balancing loads 409
multiprocessor machines
switches 10
MYIP communication parameter 404

MySybase
    accessing 360
    online support 360

N

naming conflicts 319
NBit 177
NDIS
    drivers 340
net.cfg file 342
NetBIOS
    server configuration 392
NetWare
    network adapter settings 342
network adapters
    drivers 340
network communications
    command line switches 392
network number
    in IPX address 396
network protocols
    troubleshooting 339
NEWID
    default 203
newline
    WD index delimiter 455
NodeType
    connection parameter 387
NOT NULL constraint 196
notification messages 346
notify count 191
Novell client software 340
NULL
    default 203
    on multicolumn HG index 180, 461
NULL value
    in multicolumn HG index 180, 461

O

ODBC
    configuring data sources 82
    connection parameters 363
    data sources 79
    initialization file for UNIX 84
    UNIX support 84
ODBC connectivity
    about 58
ODBC data sources
    using with jConnect 58
ODBC translation driver
    ODBC configuration 83
ODI drivers 340
offline
    dbspaces 416
online
    dbspaces 416
Open Database Connectivity
    See ODBC
option value
    truncation 496
options
    cursors 411
    DEDICATED_TASK 412
    duration 411
    MAX_TEMP_SPACE_PER_CONNECTION 115
    precedence 411
    QUERY_TEMP_SPACE_LIMIT 115
    scope 411
    setting 496
out of disk space
    monitoring space usage 331
    recommended actions 322, 323, 328
output options, monitor
    -contention 246

P

packages
    jConnect 60
page size 113
    catalog 26
    switch 26
parallel CREATE INDEX 191
partition limit 13
partitioning
    DDL Operations 163
    DML Operations 163
    example 165
    examples 166, 167
    hash partitions 166
    hash-partition-key 166
    hash-partitioning-scheme 166
    hash-range partitions 167
    hash-range-partitioning-scheme 167
    partition-key 165
    partitioning keys 163

range partitions 165
range-partitioning-scheme 165
restrictions 165, 166
SUBPARTITION BY RANGE 167
tables 163
VALUES clause (range parttitions) 165
partitions
dropping 426
read-only 143
Password connection parameter 387
password security 110
passwords
minimum length 413
path names
for databases 111
paths
relative 443
performance
effect of constraints 196
impact of table-level versioning 237
JDBC drivers 61
joins 194
queries and loads 335
permissions
procedures 256
raw devices 109
user-defined functions 271
views 154
physical layer
troubleshooting 342
ping
TCP/IP 341
PORT communication parameter 406
port number
for the database server 406
preallocating space
catalog stores 121
PreFetchOnOpen communication parameter
description 405
PREPARE statement
remote data access 254
primary keys
AUTOINCREMENT 202
creating 161, 162
multicolumn 208
privileges
command-line switches 21
defining database objects 2
problems
reporting 356
procedures
about 255
calling 256
cursors 258
dropping 256, 484
execution permissions 256
owner 257
parameters 257
result sets 257
returning results 280
returning results from 256
SQL statements allowed in 277
structure 277
product support 356
projections
optimizing 177
protocols
switch 27
troubleshooting 339
PWD connection parameter 387

Q

queries
concurrency rules 235
limiting concurrent 15
performance 180
performance issues 335
range predicates 175
thrashing 336
with DATEPART 173
query plan
index recommendations 192
query types
index types for 185
QUERY_TEMP_SPACE_LIMIT option 115

R

range partitions
examples 165
partition-key 165
range partitions 165
range-partition-decl 165
range-partitioning-scheme 165
restrictions 165
VALUES clause 165

raw devices 2
  add dbspace 109
  create database 109
  naming 443
  naming on Windows 112
  permissions 109
  setting permissions on Windows 128
  setting up access on Windows 127
  symbolic links on Windows 130
  viewing permissions on Windows 128
rawaccedit utility 127
rawaccess utility 128
RCVBUFSZ communication parameter
  description 405
read only
  locking tables 492
read-only dbspaces 142
ReceiveBufferSize communication parameter
  description 405
recovery
  from system failure 325
  server 317
  system 251
  transaction log in 251
  transactions in 251
recycling the server 35
Redirect 385
REFERENCES clause 426
referential integrity
  column defaults 197
  declaring 208
  enforcing 207
  enforcing with existing unenforced foreign keys 211
relative paths 443
RELEASE SAVEPOINT statement 249
remote data access 420
remote servers
  transaction management 254
REMOTEPWD
  using 61
repair
  database 317
request log file 354
  connection information 359
  using sa_get_request_profile 354
  using sa_get_request_times 354
request logging level 359
request-level logging 319
resource planning
  iqnumbercpus switch 19
RESTRICT action 467
result sets
  procedures 257
Rigndael encryption algorithm
  CREATE DATABASE statement 443
RLV dbspace
  file, adding 138
  file, dropping 138
rollback 217
  out-of-dbspace condition 115, 328
ROLLBACK statement 249
  syntax 494
ROLLBACK TO SAVEPOINT statement 249
root logical server policy 424
routers
  broadcasting over 396
row lock 243
row-level snapshot versioning 239

S

sa_conn_info 240
sa_get_request_profile
  analyzing request log file 354
sa_get_request_times
  analyzing request log file 354
sa_report_deadlocks system procedure 227
SAP Sybase IQ
  stopping 35
SAVEPOINT statement
  and transactions 249
  syntax 495
savepoints
  within transactions 249
scalar value subqueries 153
schedules 301
schemas
  changing 155
  creating 155
scrollable cursors
  JDBC support 61
scrolling
  cursors 253
secondary server
  shared system temporary store 123
security
  auditing 315
  Encryption [ENC] connection parameter 378

integrated logins 98, 99
minimum password length 413
SELECT * 426
SELECT DISTINCT projection 185
SELECT statement
restrictions for view creation 153
semaphores 334
SendBufferSize communication parameter
description 406
sensitivity
cursor behavior 253
separators
in WD index 455
server
CPU usage 323
deadlock 324
naming conflict 319
out of space 323
problems with shutdown 325
recovery 317
stops processing 322
transaction log 318
unique name 319
unique port number 320
unresponsive 322–324
See also database server
server log file 33
ServerName connection parameter 377
ServerPort communication parameter 406
servers
automatic startup on boot 4
connecting 58
creating logical 463
deleting logical 487
naming 9
recycling 35
stopping 35
Service Manager
starting servers 6
SESSIONS communication parameter 407
SET OPTION statement
syntax 496
SET TEMPORARY OPTION statement
syntax 496
setting dbspaces online 416
shared memory
semaphores 334
shared temporary store usage 123
shutdown
database 40
troubleshooting 325
snapshot versioning 217
row-level 239
table-level 234
SNAPSHOT_VERSIONING 239
SNDBUFSZ communication parameter
description 406
sp_iqcheckoptions stored procedure 115
sp_iqconnection 240
sp_iqdbspace
dbspace usage information 134
sp_iqdbspaceinfo
dbspace usage information 134
sp_iqestdbspaces
estimating dbspace requirements 130
sp_iqestspace
estimating database space requirements 130
sp_iqindex stored procedure 192
sp_iqindexinfo
displaying index information 136
sp_iqlocks 240
sp_iqprocedure
information about procedures 257
sp_iqprocparm
procedure parameters 257
sp_iqstatus
sample output 343
use in troubleshooting 343
sp_iqtransaction 245
determining blocking writers 337
sp_iqtransaction system procedure 218
space management
IQ main store 115
IQ temporary store 115
out-of-dbspace condition 115, 328
wait-for-space condition 115, 328
specifying
drivers 58
SQL Anywhere 16 JDBC driver
choosing a JDBC driver 61
SQL Anywhere JDBC driver
connecting to IQ and SQL Anywhere databases 58
SQLCONNECT environment variable
connections 52
srvlog
correlating connection information 359

srvlog file
connection information 359
stack trace
generating for threads 324
location 325
Start parameter
embedded databases 46
start_iq
command will not run 321
parameters 321
troubleshooting 321
start_iq utility 5
starting databases
jConnect 61
StartLine connection parameter 390
startup
troubleshooting hints 318
startup parameters 6
startup script 5
startup utility 5
STOP DATABASE statement
syntax 498
STOP ENGINE statement
syntax 499
stopping
servers 35
stopping databases 498
stored procedures
displaying information about 257
string and number defaults 203
strings
length for database options 496
strong encryption
CREATE DATABASE statement 443
Encryption [ENC] connection parameter 378
subqueries
scalar value 153
subtransactions
and savepoints 249
SUM function 181
Sybase Control Center
creating dbspaces 141
starting server 4
symbolic links 443
SYSCOLUMN table
integrity 215
SYSFOREIGNKEY table
integrity 215
SYSTABLE table
integrity 215
SYSTEM dbspace 119
system failure
recovering from 325
system procedures
sp_iqtransaction 218
system tables
about 163
system unresponsive 322
system views
integrity 215
SYSVIEW view
view information 155

T

tab
WD index delimiter 455
table constraints 467
table lock
viewing 240
table lock contention
managing 246
table partitions
DDL Operations 163
definition 163
DML Operations 163
hash partitions 166
hash-partition-key 166
hash-partitioning-scheme 166
hash-range partitions 167
partitioning keys 163
range partitions 165
table-level snapshot versioning 234
tables
adding keys to 161, 162
altering 160, 426
altering definition 426
blocked access 337
creating 155, 156, 467
dropping 161, 484
GLOBAL TEMPORARY 467
locking 234, 492
managing blocked access 337
read-only 143
temporary 482
TCP/IP
BroadcastListener [BLISTENER]
communication parameter 394

connecting across firewalls 394
LDAP communication parameter 399
server configuration 392
server port number 406
testing 341
troubleshooting 341
TDS communication parameter 408
Technical Support
checklist 361
MySybase 360
online help 360
reporting problems to 356
Telnet
TCP/IP testing 341
TEMP_DATA_IN_SHARED_TEMP
logical server policy option 424
temporary dbfiles 343
temporary storage
option to save space 253
temporary store 123
space management 115
temporary tables 482
about 157
creating 467
loading 157
versioning 245
TEXT index
creating 182
TEXT indexes 182
thrashing
HASH_THRASHING_PERCENT option 336
threads
generating a stack trace 324
increasing 16
not enough 333
TIME index 183
additional indexes 177
advantages 183
comparison to other indexes 177
disadvantages 183
recommended use 183
restrictions 183
TIMEOUT communication parameter 409
TLV 234
TO communication parameter 409
Tokenized FP (NBit) 177
trace
generating for threads 324
Transact-SQL
COMMIT TRANSACTION 441
transaction log
in system recovery 251
transaction management 254, 441
in Transact-SQL 441
transaction processing
about 217
transactions 217
about 217
blocking 226
committing 441
cursors in 252
in recovery 251
managing 254
remote data access 254
ROLLBACK statement 494
rolling back 250
SAVEPOINT statement 495
savepoints 249
starting 217
subtransactions and savepoints 249
translation driver
ODBC configuration 83
troubleshooting 317
common problems 342
database connection 326
dbisql 328
processing issues 335
protocols 339
resource issues 328
server operation 318
wiring problems 342

U

UID connection parameter 391
UNC connection parameter 391
Unconditional connection parameter 391
unenforced foreign keys 211
unique
constraint 467
UNIQUE constraints 204
unique indexes 455
UNIX
ODBC support 84
URL
jConnect 60
URL database
JDBC 60

USER special value
    default 201
user-defined data types
    CHECK conditions 206
    CREATE DOMAIN statement 453
    dropping 484
user-defined functions
    calling 269
    creating 263
    execution permissions 271
    using 263
Userid connection parameter 391
using column defaults 197
Utilities statement 488

V

VERIFY communication parameter 409
VERIFY_PASSWORD_FUNCTION option 413
versioning 217, 234, 239
    cursors and 252
    in system recovery 251
    isolation levels 221
    performance impact 237
    temporary tables 245
views
    altered tables in 426
    creating 152
    deleting 153, 155, 484
    differences from permanent tables 152
    dropping 484
    indexes 459
    inserting and deleting 153
    modifying 154
    permissions 154
    SELECT statement restrictions 153
    using 153
    working with 151
Visual Basic
    ODBC configuration for 83

W

WarehouseArchitect
    about 151
WD index 170
    advantages 172
    CHAR columns 459
    delimiters 455
    disadvantages 172
    recommended use 171
WHERE clause
    impact on index choice 185
Windows Service 4
wired memory
    setting iqwmem switch 16
wiring
    troubleshooting 342
WORD index 170
write-intent lock 244
    viewing 240, 243

Z

zr log file 354