System Administration Guide: Volume 1

Sybase IQ 15.3

DOCUMENT ID: DC00170-01-1530-01
LAST REVISED: May 2011

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Overview of Sybase IQ System Administration 1
Audience 1
Sybase IQ Design and Purpose 1
Sybase Database Architecture 1
Rapid Access to Many Data Sources 1
Data Warehousing and Sybase IQ 2
Tools for System Administration 2
The Database Server 3
Data Storage in Sybase IQ 3
SQL Anywhere and Sybase IQ 4
Security Overview 4
Concurrent Operations 5
Multiplex Capability 5
Stored Procedures 6
System Tables and Views 7
Commands and Functions 7
Types of SQL Statements 7
Functions 8
Message Log Contents 8
Version Information in Message Logs 9
Message Log File Management 11
The Utility Database 14
Scalability 15
Memory Use 15
Data Loads 15
Processing Threads 16
Disk Space 16
Intermediate Versioning 17
Page Size 17
Column-based Indexes 17
Query Optimizer 17

Schema Design 17
UNION ALL Views 18
Sybase IQ Startup 19
Ways to Start Database Servers 19
The Server Startup Utility 20
Starting Servers from the Windows Start Menu
....21
The Server as a Windows Service 21
Automatic Server Startup 21
Command Line Switches 21
Displaying Command Line Options 22
Configuration Files 22
Command Line Example Formatting in Sybase
IQ Documentation 23
Required Command Line Options 23
Default Configuration File 24
Configuration File for the Demo Database 24
Naming Restrictions 24
Server and Database Names 24
Command-line Options for Performance 26
Command Line Options that Control
Permissions 30
Maximum Catalog Page Size 31
Client/Server Environment Options 31
Forced Recovery Mode Options 33
Starting a Server from Interactive SQL 33
Shared Memory Conflicts 34
Server Activity Logs 35
Naming the Server Log File 38
UNIX Log Files 38
Database Server Shutdown 38
When to Stop and Restart the Server 39
Ways to Stop Database Servers 39
Who Can Stop the Server 41
Operating System Session Shutdown 42

Ways to Start and Stop Databases 42
Database Startup Guidelines 43
Ways to Stop Databases 44
How to Start the iqdemo Database 44
Sybase Central Start and Stop Methods 44
Sybase IQ Connections 45
Connection Status 45
How Sybase IQ Establishes Connections 46
Learning Roadmap for Connections 46
How Connection Parameters Work 47
Format for Connection Strings 48
How Applications Pass Connection Parameters 48
Connection Parameters in ODBC Data Sources 49
Sybase Central or Interactive SQL Connections 49
The Connect Dialog 50
Opening the Connect dialog (Sybase Central) 50
Opening the Connect dialog (Interactive SQL) 51
Drivers for Connections 51
Connection Shortcuts in Sybase Central 52
Server Objects 52
Creating a Connection Profile 52
Simple Connection Examples 53
Connecting to the demo database from Sybase Central 53
Connecting To the Demo Database From Interactive SQL 54
How Database Status Affects Local Connections 55
Connecting To a Running Database On a Local Server 55
Connecting to a database that is not yet running 56
Connecting to a database from Interactive SQL on UNIX 56
Connecting from a UNIX system 57

Connecting From a Windows System 58
Connections to Embedded Databases 59
Connecting Using a Data Source 60
Default Connection Parameters 61
Connecting From Sybase IQ Utilities 62
ODBC Data Sources 62
Where Data Sources Are Held 63
Creating a Data Source from the ODBC Administrator 64
Creating an ODBC Data Source from the Command Line 65
Testing an ODBC data source 65
Configuring ODBC Data Sources in ODBC Administrator 66
ODBC Tab 66
File Data Sources 67
Creating a File Data Source Using the ODBC Administrator 68
File Data Sources and Text Editors 68
ODBC Data Sources on UNIX 68
The iAnywhere Solutions Oracle Driver 70
Creating an Oracle DSN on UNIX 71
Creating an Oracle DSN on Windows 72
Creating an Oracle DSN Using IQDSN 72
Database Connections Using OLE DB 72
OLE DB Providers 73
Connections from ADO 73
Connections From Other Databases 74
Avoiding Port Number Conflicts on UNIX 75
How to Test Connections 75
Integrated Logins 76
Using Integrated Logins 77
Security Concerns: Unrestricted Database Access 78

Temporary Public Options Provide Added Security 79
Network Aspects of Integrated Logins 80
Default Integrated Login Users 80
Logical Server Configuration 81
Connections in Simplex 81
Connections in Multiplex 81
How to End Connections 82
Connection Logging 82
Connection and Communication Parameters 83
Connection Parameters 83
AppInfo Connection Parameter [Appinfo] 84
AutoPreCommit Connection Parameter [AutoPreCommit] 86
AutoStart Connection Parameter [Astart] 86
AutoStop connection parameter [Astop] 87
CharSet connection parameter [CS] 88
CompressionThreshold Connection Parameter [COMPTH] 88
CommBufferSize Connection Parameter [CBSize] 88
CommBufferSpace connection parameter [CBSpace] 89
CommLinks Connection Parameter [Links] 90
ConnectionName Connection Parameter [CON] 91
DatabaseFile Connection Parameter [DBF] 92
DatabaseName Connection Parameter [DBN] 93
DatabaseSwitches Connection Parameter [DBS] 94
DataSourceName Connection Parameter [DSN] 95
DBKEY Connection Parameter [DBKEY] 95
DisableMultiRowFetch Connection Parameter [DMRF] 96

EngineName Connection Parameter [ENG] 96
EncryptedPassword Connection Parameter [ENP] 97
Encryption Connection Parameter [ENC] 98
FileDataSourceName Connection Parameter [FileDSN] 99
Idle Connection Parameter [IDLE] 100
Integrated Connection Parameter [INT] 101
Language Connection Parameter [LANG] 102
LazyClose Connection Parameter [LCLOSE] 102
LivenessTimeout Connection Parameter [LTO] 103
LogFile Connection Parameter [LOG] 104
NewPassword Connection Parameter [NEWPWD] 105
Password Connection Parameter [PWD] 105
PrefetchBuffer Connection Parameter [PBUF] ...105
PrefetchRows Connection Parameter [PROWS] 106
Prompt Connection Parameter [PROMPT] 107
RetryConnectionTimeout Connection Parameter [RetryConnTO] 107
ServerName connection parameter [ENG] 107
StartLine Connection Parameter [START] 108
Unconditional Connection Parameter [UNC] 108
Userid Connection Parameter [UID] 109
Network Communications Parameters 109
Broadcast Communication Parameter [BCAST] 111
BroadcastListener Communication Parameter [BLISTENER] 111
Certificate Communication Parameter 112
Certificate_Password Communication Parameter 112
ClientPort Communication Parameter [CPort] ...113

DatabaseName Communication Parameter [DBN] ... 114
DoBroadcast Communication Parameter [DBROAD] ... 115
Host Communication Parameter [IP] ... 116
LDAP Communication Parameter [LDAP] ... 117
LocalOnly Communication Parameter [LOCAL] ... 118
LogFile Communication Parameter [LOG] ... 119
LogFormat Communication Parameter [LF] ... 119
LogMaxSize Communication Parameter [LSIZE] ... 120
LogOptions Communication Parameter [LOPT] ... 120
MaxConnections Communication Parameter [MAXCONN] ... 121
MaxRequestSize Communication Parameter [MAXSIZE] ... 122
MyIP Communication Parameter [ME] ... 122
PreFetchOnOpen Communication Parameter ... 123
ReceiveBufferSize Communication Parameter [RCVBUFSZ] ... 124
SendBufferSize Communication Parameter [SNDBUFSZ] ... 124
ServerPort Communication Parameter [PORT] ... 125
Sessions Communication Parameter ... 126
TDS Communication Parameter ... 127
Timeout Communication Parameter [TO] ... 127
VerifyServerName Communication Parameter [Verify] ... 128
Database Object Management ... 131
Sybase IQ Database Design ... 131
Tools for Working with Database Objects ... 131
Sybase Central ... 132
Interactive SQL ... 132

Setting Up a Sybase IQ Database 133
Guidelines for Scheduling Data Definition Tasks 133
Dummy Tables for Performance Monitoring 134
Authorities for Data Definition 134
Device Selection 134
Space Allocation 135
Range Partitions 143
Restrictions on DDL Operations on Partitions 144
DML Operations on Partitions 145
Object Placement for Non-Partitioned Tables 145
Object Placement for Partitioned Tables 146
Database Definition 147
Database Creation with SQL 147
Setting Database Options in Sybase Central 157
Setting Database Options in Interactive SQL 157
IQ Main Store and IQ Temporary Store Space Management 157
Showing System Objects in Sybase Central 159
Showing System Objects in Interactive SQL 159
Disconnecting from a Database in Sybase Central 159
Disconnecting All Connections from a Database in Interactive SQL 159
Disconnecting from a Database in Embedded SQL 160
Disconnecting Other Users From a Database 160
Dropping a Database 161
Data Storage 161
Dbfile Attributes and Operations 161
Dbspace Attributes and Operations 162
Read-only and Read-write Dbspaces and Files 162
Dbspace Renaming Guidelines 169
Additional Dbspaces 169
Guidelines for Dropping a Dbspace 171

Dbspace Management Example 172
Table Management 179
Guidelines for Creating Tables 179
Guidelines for Altering Tables 185
Guidelines for Dropping Tables 186
Creating Primary Keys 186
Creating Foreign Keys 187
Table Information in System Views 188
View Management 188
Creating Views 189
Guidelines for Using Views 190
Guidelines for Modifying Views 190
Permissions on Views 191
How to Delete Views 191
View Information in System Views 192
Index Management 192
Introduction to Indexes 192
Creating Indexes 193
Index Information in System Views 193
Index Validation 194
How to Rename Indexes 194
How to Remove Indexes 194
Sybase IQ Indexes 195
Overview of Indexes 195
Sybase IQ Index Types 195
Benefits Over Traditional Indexes 198
Tools for Creating Sybase IQ Indexes 198
Interactive SQL Index Creation 199
Concurrent Column Index Creation 200
Criteria for Choosing Indexes 200
Number of Unique Values in the Index 200
Types of Queries 201
Disk Space Usage 203
Data Types in the Index 204
Combining Index Types 205

Sybase IQ Index Types 205
The Fast Projection (FP) Default Index Type 205
The Low_Fast (LF) Index Type 209
The High_Group (HG) Index Type 211
The High_Non_Group (HNG) Index Type 212
The Compare (CMP) Index Type 213
The Containment (WD) Index Type 214
The Date (DATE), Time (TIME), and Datetime (DTTM) Index Types 216
TEXT Indexes 220
Optimizing Performance for Ad Hoc Joins 221
Selecting an Index 221
Adding Column Indexes After Inserting Data 222
Using Join Indexes 223
Join Indexes Improve Query Performance 223
Loading Considerations for Join Indexes 223
How Join Indexes are Used for Queries 223
Relationships in Join Indexes 223
When a Join Becomes Ad Hoc 224
Join Hierarchy Overview 224
Columns in the Join Index 225
The Join Hierarchy in Query Resolution 225
Multiple Table Joins and Performance 227
Steps in Creating a Join Index 228
Synchronizing Join Indexes 229
Defining Join Relationships Between Tables 230
Issuing the CREATE JOIN INDEX Statement 233
Creating a Join Index in Sybase Central 233
Types of Join Hierarchies 234
Restrictions on Modifying Join Index Tables 237
Insertions or Deletions from Join Index Tables ..237
Table Versioning Controls Access to Join Indexes 238
Size and Benefits of Join Indexes 239
How to Estimate Join Index Size 239

How to Determine Join Index Benefits 239
Data Import and Export 241
Import and Export Overview 241
Import and Export Method Selection 241
Input and Output Data Formats 242
Permissions for Modifying Data 243
Schedule Database Updates 243
Methods for Exporting Data from a Database 244
Output Redirection 244
Data Extraction Facility 245
Bulk Loads with the LOAD TABLE Statement 254
Loads that Specify Input Data Format 255
Direct Loading of Data from Clients 258
Considerations for Partitioned Table Loads 258
Load and Insert Messages 259
Integrity Constraint Violation Messages 259
Binary Load Formats 263
IQ Binary Load Format and Load Efficiency 263
Operating System Native Data Types 263
DATE 264
TIME 264
TIMESTAMP 264
NUMERIC and DECIMAL 265
NULL Value Loads 266
Using the INSERT Statement 268
Inserting Specified Values Row by Row 269
Inserting Selected Rows from the Database 270
Inserting from a Different Database 270
Interactive Data Imports 272
Moving Data Between Systems with Different Endian Formats 273
Inserting into Tables of a Join Index 274
Insertions into Primary and Foreign Key Columns 275
Load or Extraction of Large Object Data 275
Data Conversion on Insertion 276

Data from Pre-Version 15 Sybase IQ 277
Load Conversion Options 277
Column Width Issues 281
Faster Date and Time Loads 282
ASCII Input Conversion 283
The DATE Option 285
The DATETIME Conversion Option 287
NULL Data Conversions 290
Other Factors Affecting the Display of Data 290
Matching Adaptive Server Enterprise Data Types 291
Unsupported Adaptive Server Enterprise Data Types 291
Adaptive Server Enterprise Data Type Equivalents 292
Conversion Errors on Data Import 297
Tune Bulk Loading of Data 297
Load Performance During Database Definition 297
Load Time Environment Adjustments 298
IQ Main Store Space Use in Incremental Loads 299
Thread Use During Loads 299
Changes to Table Rows 300
Ways to Delete Data 301
Managing User IDs and Permissions 303
Database Permissions and Authorities Overview 303
Inheriting Authorities 304
Inherited Permissions 305
Authorities Overview 305
Ownership Permissions Overview 308
Dbspace Management Permissions 309
Table and Views Permissions Overview 309
Group Permissions Overview 310
Server Command-Line Permission Options 310
Login Management 311

User Account and Connection Management 311
Preventing Connection After Failed Login Attempts 313
Locking Out Users 313
Unlocking Users 313
Utility Database Server Security 314
Starting the Utility Database 314
Defining the Utility Database Password 315
Permission to Execute File Administration Statements 315
Managing Individual User IDs and Permissions 316
Creating a User with Interactive SQL 317
Creating a User in Sybase Central 317
Creating a User in Sybase Control Center 317
Changing a Password 318
Granting DBA Authority to a User 319
Granting OPERATOR Authority to a User 319
Granting PERMS ADMIN Authority to a User 319
Granting RESOURCE Authority to a User 319
Granting SPACE ADMIN Authority 320
Granting USER ADMIN Authority to a User 320
Permissions on Tables and Views 320
Granting the Right to Grant Permissions 322
Granting Permissions on Procedures in Interactive SQL 323
Revoking User Permissions in Interactive SQL 324
Group Management 325
Creating Groups in Interactive SQL 325
Creating Groups in Sybase Central 326
Creating Groups in Sybase Control Center 326
Adding Group Members in Interactive SQL 326
Adding Group Members in Sybase Central 327
Adding Group Members in Sybase Control Center 327

Permissions of Groups 327
Tables Owned by Groups 328
Groups Without Passwords 328
Groups Created Automatically 329
Owner Prefixes for Object Names 330
Views and Procedures Provide Extra Security 332
Views Provide Tailored Security 332
Procedures Provide Tailored Security 334
How User Permissions are Assessed 335
Resources Used by Connections 336
Database Options that Govern User Resources 336
Limits on Database Connections 337
Using Procedures to Disable Connections 337
Users and Permissions in System Objects 337
User ID, Group and Permissions Information in System Tables 338
User ID, Group, and Permissions Information in System Views 338
Transport-Layer Security 339
IPv6 Support 339
Data Integrity 341
Data Integrity Overview 341
How Data Can Become Invalid 341
Rules and Checks for Valid Data 341
Statements that Change Database Contents 343
Data Integrity Tools 343
SQL Statements for Implementing Integrity Constraints 344
Column Defaults Encourage Data Integrity 344
Supported Default Values 344
Default Value Restrictions 345
Creating Column Defaults 345
Changing Column Defaults 346
Deleting Column Defaults 346

Supported Column Default Values 346
Column Defaults in Sybase Central 348
Date, Time, and Timestamp Defaults 348
USER Defaults 349
The IDENTITY or AUTOINCREMENT Default 349
The NEWID Default 350
The NULL Default 350
String and Number Defaults 350
Constant Expression Defaults 351
Table and Column Constraints 351
UNIQUE Constraints on Columns or Tables 351
IQ UNIQUE Constraints on Columns 352
CHECK Conditions on Columns 352
CHECK Conditions on User-Defined Data Types 353
Adding, Altering and Deleting Column Constraints in Sybase Central 353
CHECK Conditions on Columns 354
CHECK Conditions on Tables 355
Removing Check Conditions on Tables Using Interactive SQL 355
Entity and Referential Integrity 355
How to Declare Entity Integrity 355
How to Enforce Entity Integrity 356
If a Client Application Breaches Entity Integrity 356
Referential Integrity 357
Loss of Referential Integrity 360
Concurrent Operations 361
How to Disable Referential Integrity Checking 363
Integrity Rules in the System Tables 363
Transactions and Versioning 365
Transactions and Versioning Overview 365

Transactions 365
Concurrency Overview 367
Versioning Overview 368
Versioning Prevents Inconsistencies 376
How Locking Works 376
Locks for DML Operations 377
Locks for DDL Operations 378
Primary Keys and Locking 379
Tools for Managing Locks 379
Displaying Active Locks 379
Tools for Investigating Lock Contention 380
Isolation Levels 381
Checkpoints, Savepoints, and Transaction Rollback 382
Checkpoints 382
Savepoints Within Transactions 383
Transaction Rollback 385
System Recovery 385
How Transaction Information Aids Recovery 386
Performance Implications of Snapshot Versioning 387
Overlapping Versions and Deletions 387
Cursors in Transactions 388
Cursors and Versioning 389
Cursor Sensitivity 389
Cursor Scrolling 389
Hold Cursors 390
Positioned Operations 390
How to Control Message Logging for Cursors 390
International Languages and Character Sets 391
Default Collation 392
Understanding Character Sets in Software 392
Code pages in Windows 392
How the Collation Sequence Sorts Characters 393

Understanding Locales 394
Understanding Collations 394
Displaying Collations 394
ANSI and OEM Code Pages 395
ANSI ISO_1 Collation 395
ANSI 1252LATIN1 Collation 395
ANSI ISO1LATIN1 Collation 395
ANSI ISO9LATIN1 Collation 396
Multibyte Collations 396
Understanding Character Set Translation 397
Character Translation for Database Messages 397
Connection Strings and Character Sets 398
Avoiding Character-Set Translation 398
Configuring Your Character Set Environment 399
Locale Information 400
Setting a Locale 400
Setting the Locale for an INSERT...LOCATION Statement 401
Creating a Database with a Named Collation 401
Disabling Character Set Translation on a Database Server 402
Changing a Database From One Collation to Another 402
Compatibility Issues 402
Performance Issues 402
Data Backup, Recovery, and Archiving 405
Data Protection 405
How to Back Up Databases 406
Types of Data Stores 406
Types of Backups 406
Select Archive Devices 409
Preparing for Backup 410
Running Backups 412
Specify Operator Presence 412

Specify the Type of Backup 413
Specifying Virtual Backup 414
Specifying Archive Devices 414
Other Backup Options 417
Wait for Tape Devices 418
Backup and Restore Using Read-Only Hardware 418
Backup Examples 418
Recovery from Errors During Backup 419
After you Complete a Backup 420
Performing Backups with Non-Sybase Products 420
Virtual Backups 421
Types of Virtual Backups 421
Virtual Backup with SAN Snapshot or Shadow Hardware 423
System-Level Backups 423
Shut Down the Database 423
Back Up the Right Files 424
Restoring from a System-Level Backup 425
Validating Your Database 426
Restoring Your Databases 427
Before You Restore 427
The RESTORE Statement 430
Restoring in the Correct Order 434
Reconnecting After You Restore 436
Renaming the Transaction Log after you Restore 437
Validating the Database After You Restore 437
Restore Requires Exclusive Write Access 437
Displaying Header Information 438
Recovery from Errors During Restore 438
Verifying a Database Backup 439
Backups and Symbolic Links (UNIX Only) 441
Getting Information about Backups and Restores 441

Locating the Backup Log 442
Content of the Backup Log 442
Maintaining the Backup Log 443
Recording dbspace Names 443
Determining Your Data Backup and Recovery Strategy 445
Scheduling Routine Backups 445
Designating Backup and Restore Responsibilities 446
Improving Performance for Backup and Restore 446
Archiving Data with Read-Only Hardware 448
Using Read-Only Hardware 448
System Recovery and Database Repair 453
Recovery and Repair Overview 453
Normal Recovery 454
Database Verification 454
The sp_iqcheckdb Stored Procedure 455
sp_iqcheckdb Output 458
Resource Issues Running sp_iqcheckdb 461
Database Repair 462
Analysis of Index Errors 462
Index Error Repair 465
Analysis of Allocation Problems 466
Repairing Allocation Problems using DBCC 468
Forced Recovery Mode 470
Starting a Server in Forced Recovery Mode 471
Recovering Leaked Space 472
Recovering Multiplex Databases 474
Problems Reported by DBCC 474
Index Problems that DBCC Cannot Repair 475
Dropping Inconsistent Indexes, Tables, or Columns 475
DBCC Error Messages 476
Troubleshooting Hints 481

Sources of Online Support 481
Solutions for Specific Conditions 481
Decision Flow for Server Recovery and Database Repair 481
Server Operational Issues 482
Database Connection Issues 492
Interactive SQL (dbisql) Issues 494
Resource Issues 495
Processing Issues 504
Performance Issues 508
Sybase Central Issues 509
Troubleshooting Network Communications 509
Ensuring that You are Using Compatible Protocols 509
Ensuring that You Have Current Drivers 510
Switching off Your Computer Between Reboots 510
Diagnosing Your Protocol Stack Layer by Layer 510
Testing a TCP/IP Protocol Stack 510
Diagnosing Wiring Problems 512
Checking Common Network Communications Problems 512
Diagnostic Tools 513
Restoring to a New Temporary File Topology 513
The sp_iqstatus Stored Procedure 513
Interpreting Notification Messages 515
The sp_iqcheckdb Stored Procedure 520
Checking Database and Server Startup Option Values 521
Finding the Currently Executing Statement 521
Logging Server Requests 522
Connection for Collecting Diagnostic Information 525
Diagnosing Communications Issues 525

Reporting Problems to Technical Support 525
Collecting Diagnostic Information Using getiqinfo 525
Information Collected by getiqinfo 527
Correlating Connection Information Between the .srvlog and .iqmsg Files 528
Another Source of Helpful Information 529
Checklist: Information for Technical Support 530
Index 533

Overview of Sybase IQ System Administration

Understanding Sybase® IQ design and functionality provides a foundation for system administration.

Audience

Sybase IQ system administrators manage database setup, connections, loading, security, backup, and troubleshooting.

Topics on managing a Sybase IQ system are intended for database administrators and others who need to understand database creation, load, and operation issues.

Sybase IQ Design and Purpose

Sybase IQ is a high-performance decision support server designed specifically for data warehousing.

This cross-platform product runs on several popular Unix, Linux, and Windows platforms.

Sybase IQ is part of the Adaptive Server® family that includes *Adaptive Server Enterprise* for enterprise transaction and mixed workload environments and *SQL Anywhere*, a small footprint version of Adaptive Server often used for mobile and occasionally connected computing.

Sybase Database Architecture

Sybase database architecture provides a common code base for Sybase IQ and SQL Anywhere, with workload optimized data stores.

You use the IQ store for data warehousing. These products share a common command syntax and user interface, allowing easier application development and user access.

Rapid Access to Many Data Sources

Sybase IQ can integrate data from diverse sources—not just IQ databases, but other databases in the Adaptive Server family, as well as non-Sybase databases and flat files.

You can import this data into your IQ database to take advantage of IQ's rapid access capabilities. You can also query other databases directly, using Sybase IQ's remote data access capabilities.

Data Warehousing and Sybase IQ

Data warehouses are collections of data designed to allow business analysts to analyze information.

Typically, data warehouses are distinct from production databases, to avoid interrupting daily operations. Data warehouses are often used as data stores on which to build decision support systems (DSS). A *decision support system* is a software application designed to allow an organization to analyze data in order to support business decision making.

All of Sybase IQ's capabilities are designed to facilitate DSS applications. A unique indexing system speeds data analysis. Query optimization gives you rapid responses, even when results include thousands or millions of rows of data. Concurrent data access for multiple query users, and the ability to update the database without interrupting query processing, provide the 24–hour, 7–day access that users expect.

Sybase IQ *multiplex* is a highly scalable shared disk grid technology that allows concurrent data loads and queries via independent data processing nodes connected to a shared data source. For details and syntax, see *Using Sybase IQ Multiplex*.

Tools for System Administration

Tools for managing Sybase IQ databases are Sybase Central, Sybase Control Center, and Interactive SQL.

Sybase Central is a graphical application for managing database objects. You can use Sybase Central to perform common administrative tasks such as creating databases, backing up databases, adding users, adding tables and indexes, enabling databases for multiplex capability, and monitoring database performance. Sybase Central has a Java-based graphical user interface, and can be used with any operating system that allows graphical tools.

Interactive SQL is a utility that allows you to enter SQL statements interactively and send them to a database, or execute them as scripts. Interactive SQL has two forms: Java-based **dbisql** and the Open Client utility **iqisql**.

The *Introduction to Sybase IQ* explains how to use Sybase Central and Interactive SQL to perform simple administrative tasks. If you are not already familiar with these tools, you should read about them in the *Introduction to Sybase IQ* and use the tutorials provided there.

Sybase Control Center for Sybase IQ is a Web-based tool for managing and monitoring Sybase IQ single-node and multiplex servers. The two main features are administration and monitoring.

The Sybase Control Center architecture allows a small number of Sybase Control Center servers to monitor all Sybase IQ servers in an enterprise using the Sybase Control Center agent.

The Sybase Control Center agent is installed with each Sybase IQ server.

See *Sybase Control Center 3.2.0 > Sybase Control Center for Sybase IQ*.

In addition to these tools, Sybase IQ provides a number of stored procedures that perform system management functions. You can also create your own procedures and batches. You may wish to take advantage of IQ's event-handling capability to develop your own system management tools.

See *System Administration Guide: Volume 2 > Automating Tasks Using Schedules and Events*.

A few administrative tasks, such as selecting a collation, rely on command-line utilities described in the *Utility Guide*.

See also

- *Stored Procedures* on page 6

The Database Server

The *database server* is the "brain" of your Sybase IQ system.

Access data through the database server, never directly. The database server receives user requests for information, connects to a specified database, and carries out the instructions.

Data Storage in Sybase IQ

Tablespaces, dbspaces, dbfiles, and stores are units in which Sybase IQ stores data.

A *tablespace* is a unit of storage within the database that you can administer as a logical subset of the total storage. Individual objects and subobjects may be allocated to individual tablespaces. Each database includes multiple tablespaces.

In Sybase IQ 15.3, a *dbspace* is a tablespace that consists of one or more operating system files.

The meaning of the term varies according to the product version you are using. Sybase IQ 12.7 implemented one-to-one mapping between a dbspace and a database file. With `DEFAULT_DISK_STRIPING` option 'ON', Sybase IQ automatically distributed data across all available dbspaces for optimal performance and ease of administration. Users could not specify in which dbspace a table or index should be created.

The term *dbfile*, with a corresponding logical file name and physical file path, refers to each operating system file.

Each physical file path must be unique. The dbfile name can be the same as the dbspace name.

A *store* is one or more dbspaces that store persistent or temporary data for a special purpose. Sybase IQ has three stores:

- The *catalog store* contains the SYSTEM dbspace and up to twelve additional catalog dbspaces.
- The *IQ main store* contains the IQ_SYSTEM_MAIN dbspace and other user defined dbspaces.
- The *IQ temporary store* contains the IQ_SYSTEM_TEMP dbspace.

See also

- *Types of Dbspaces* on page 135

SQL Anywhere and Sybase IQ

SQL Anywhere is a relational database system that can exist with or without IQ.

The catalog store closely resembles a SQL Anywhere store. 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.

SQL Anywhere tables have a different format than IQ tables. While the commands you use to create objects in a SQL Anywhere database may be the same as those for an IQ store, there are some differences in the features you can specify in those commands. Always use the command syntax in this book or Reference: Statements and Options for operations in the IQ store.

This guide explains how to manage an IQ store and its associated catalog store. If you have a SQL Anywhere database, or if you have SQL Anywhere-style tables in your catalog store, see the SQL Anywhere documentation for details of how to create, maintain, and use them.

Security Overview

Sybase IQ provides security controls by means of the privileges you can assign to users.

The database administrator (DBA) is responsible for maintaining database security.

Types of Users

Sybase IQ recognizes three categories of users for each Sybase IQ database: DBA, owner, and public users.

The database administrator, or *DBA*, has complete authority to perform all operations on that database. This guide is addressed primarily to the DBA, who typically carries out most administrative tasks.

The user who creates a particular database object is its *owner*, and can perform any operation on that object.

All other users are considered *public users*. The owner of an object is considered a public user for objects owned by other users.

Permissions for Tasks

Except for the DBA, who can perform any task, users must be granted the authority to perform specific tasks.

For example, you need the proper authority to:

- Connect to a database.
- Create database objects, such as a database, table, index, or foreign key.
- Place objects on specific dbspaces.
- Alter the structure of database objects.
- Insert, update, or delete data.
- Select (view) data.
- Execute procedures.

See also

- *Managing User IDs and Permissions* on page 303
- *Permissions for Modifying Data* on page 243
- *Transactions and Versioning* on page 365

Concurrent Operations

Sybase IQ allows multiple users to query a database at the same time, while another user inserts or deletes data, or backs up the database.

Changes to the structure of the database, such as creating, dropping, or altering tables, temporarily exclude other users from those tables, but queries that only access tables elsewhere in the database can proceed.

Sybase IQ keeps your database consistent during these concurrent operations by maintaining multiple versions of table data.

Multiplex Capability

Multiplex is a powerful feature in Sybase IQ that allows you to scale your applications by using multiple machines in a clustered system to mount and open a single IQ database residing on a shared disk storage.

For more information, see *Using Sybase IQ Multiplex*.

See also

- *CREATE DATABASE Statement Defaults* on page 150

- *Database Creation with SQL* on page 147
- *Relative Path Names* on page 151
- *Utility Database Server Security* on page 314

Stored Procedures

Sybase IQ *stored procedures* give you information about your database and users and carry out various operations on the database.

A stored procedure typically operates on the database in which you execute it. For example, if you run the stored procedure **sp_addlogin** in the `iqdemo` database, it adds a user to `iqdemo`.

Note: Statements shown in examples generally use the `iqdemo` database, a demo database that you can create using scripts installed with Sybase IQ.

You can also create your own stored procedures.

Sybase IQ Stored Procedures

Sybase IQ provides numerous stored procedures for viewing and managing information.

For example, **sp_iqstatus** displays general status information, while **sp_iqcheckdb** checks the validity of the current database and can repair allocation problems.

See *System Administration Guide: Volume 2 > System Procedures*.

Several multiplex stored procedures are also available. See *Using Sybase IQ Multiplex > Multiplex Reference > System Procedures.*

Catalog Stored Procedures

In addition to most Adaptive Server Enterprise Catalog stored procedures, there are other system and catalog stored procedures.

For a complete list, see Reference: Building Blocks, Tables, and Procedures > System Procedures.

Sybase IQ does not support Adaptive Server Enterprise Catalog stored procedures **sp_column_privileges**, **sp_databases**, **sp_datatype_info**, and **sp_server_info**.

See also

- *Tools for System Administration* on page 2

System Tables and Views

Sybase IQ system tables and views are objects that contain all of the information the database server needs to manage your Sybase IQ system. System views present the information from their corresponding system tables in a more readable format.

The system tables reside in the catalog store, and are sometimes called catalog tables. Access to system tables is only through system views. The SYS user ID owns the system tables.

Among the information in the system tables is:

- Database characteristics
- Table characteristics, including table definitions and information about the size and location of each table
- Information about indexes
- Current settings for database and Interactive SQL options

For a complete description of system tables and views and their contents, see *Reference: Building Blocks, Tables, and Procedures*.

Commands and Functions

All Sybase IQ commands are SQL statements. SQL stands for Structured Query Language, a language commonly used in database applications.

Sybase IQ SQL uses the same syntax as SQL Anywhere SQL; the only differences are for certain product capabilities that are supported only for Sybase IQ or for SQL Anywhere. Sybase IQ SQL also offers a high degree of compatibility with Transact-SQL, the SQL dialect used by Adaptive Server Enterprise.

This section introduces the types of statements and functions you can use. Other chapters of this book tell you about the statements you use to perform various administrative tasks.

For complete details of supported functions and statements, see *Reference: Building Blocks, Tables, and Procedures* and *Reference: Statements and Options*.

Types of SQL Statements

SQL statements are comprised of three types: data definition language (DDL), data manipulation language (DML), and program control statements.

DDL (data definition language) statements let you define and modify your database schema and table and index definitions. Examples of DDL statements include **CREATE TABLE**, **CREATE INDEX**, **ALTER TABLE**, and **DROP**.

DML (data manipulation language) statements let you query your data, and move data into and out of the database. Examples of DML statements include **SELECT**, **SET**, and **INSERT**.

Program control statements control the flow of program execution. They do not operate directly on your Sybase IQ tables. Examples include **IF**, **CALL**, and **ROLLBACK**.

Functions

Functions return information from the database.

Functions are allowed anywhere an expression is allowed. Sybase IQ provides functions that:

- Aggregate data (for example, **AVG**, **COUNT**, **MAX**, **MIN**, **SUM**, **STDDEV**, **VARIANCE**)
- Manipulate numeric data (for example, **ABS**, **CEILING**, **SQRT**, **TRUNCATE**)
- Manipulate string data (for example, **LENGTH**, **SOUNDEX**, **UCASE**)
- Manipulate date and time data (for example, **TODAY**, **DATEDIFF**, **DATEPART**, **MINUTES**)
- Convert retrieved data from one format to another (**CAST**, **CONVERT**)
- Manipulate analytical data (for example, **DENSE_RANK**, **NTILE**, **PERCENT_RANK**, **PERCENTILE_CONT**, **PERCENTILE_DISC**, **RANK**)

Message Log Contents

An IQ message log file exists for each database to capture messages from the server about operations.

The message log file has the default name `dbname.iqmsg`, and is created in the same directory as the catalog store when you start a newly created database. The database creator may specify a different location, a different file name, or both.

By default, Sybase IQ logs the following types of messages in the message log file:

- Error messages
- Status messages
- Insert notification messages
- Query plans

You can examine this file as you would any other text file.

At the beginning of the file, and when you start a database, you see output similar to:

```
I. 05/06 17:07:25. 0000000000 OpenDatabase Completed
I. 05/06 17:07:25. 0000000000 IQ cmd line srv opts:
I. 05/06 17:07:25. 0000000000 IQ full cmd line: -c 48m -gc 20 -gd all
-gl all -gm 10 -gp 4096 -ti 4400 -n system1_demo -x tcpip{port=1870}
iqdemo.db -gn 25 -o /system1/users/user1/sybase/IQ-15_3/logfiles/
system1_iqdemo.0007.srvlog -hn 7
I. 05/06 17:07:25. 0000000000 DB: r/w, Main Buffs=127, Temp Buffs=95,
Pgsz=131072/8192blksz/16bpc
I. 05/06 17:07:25. 0000000000 DB: Frmt#: 23F/2T/1P (FF: 03/18/1999)
I. 05/06 17:07:25. 0000000000 DB: Versn:
15.3.0.5071/090501/P/Mainline/Sun_Opteron/OS 5.10/64bit/2009-05-01
01:21:40
I. 05/06 17:07:25. 0000000000 DB: Name:  /system1/users/user1/
```

```
iqdemo.db
I. 05/06 17:07:25. 0000000000 DB: Txn ID Seq: 1
I. 05/06 17:07:25. 0000000000 DB: DBID Blk: 7730
I. 05/06 17:07:25. 0000000000 DB: IQ Server system1_demo, PID 24485,
LOGIN user1
I. 05/06 17:07:25. 0000000000 DB: Database encryption is OFF.
I. 05/06 17:07:25. 0000000000
Mem: 44mb/M44
Main      Blks: U7841/20%, Buffers: U6/L1
Temporary Blks: U65/0%, Buffers: U4/L0
Main      I: L24/P4 O: C2/D2/P0 D:0 C:100.0
Temporary I: L34/P0 O: C4/D4/P0 D:0 C: 0.0
I. 05/06 17:07:25. 0000000000 Collation ISO_BINENG, Case Respect,
Blank Padding On, Comparisons are Binary
I. 05/06 17:07:26. 0000000000 RcvyCmpl
I. 05/06 17:07:26. 0000000000 Chk
I. 05/06 17:07:26. 0000000000 ChkDone [NumTxnCP: 0]
I. 05/06 17:07:26. 0000000000 PostChk
I. 05/06 17:07:26. 0000000000 CloseDatabase
```

Version Information in Message Logs

Version information appears near the beginning of the message log file.

Version information is similar to the following:

```
Versn:
15.3.0.5071/100301/P/GA/Sun_Opteron/OS 5.10/64bit/2010-05-01
01:21:40
```

Table 1. Version string elements

Version string element	Example	Comments
Major release	15.3	First two segments indicate major release
Minor release	15.3.*n*	Third segment indicates minor release
Internal build number	5071	Unique build number for internal use
Internal build date	100301	Date identifier for internal use
Internal build type	P	P = Production build
Release type	GA	GA = General Availability ESD = Engineering Software Distribution (Maintenance)
Hardware platform	Sun_Opteron	Identifier for hardware provider
Operating system version	OS 5.10	OS version and bit mode refer to the system on which the software was built, not where it is running currently.
Bit mode	64bit	

Version string element	Example	Comments
Build date and time	2010-03-01 01:21:40	Shown as `YYYY-MM-DD hh:mm:ss` (ISO datetime format): 4-digit year 2-digit month number (0 – 12) 2-digit day of month number (01 – 31) 2-digit number of complete hours that have passed since midnight (00 – 23) 2-digit number of complete minutes that have passed since the start of the hour (00 – 59) 2-digit number of complete seconds that have passed since the start of the minute (00 – 59)

Collation Information in Message Logs

The message log captures collation information when you start the database.

Collation information captured in the message log shows the collation name, case sensitivity, blank padding state, and character translation table requirement. `Comparisons are conditioned` means that no translation table is required when comparing character data.

Connection Information in Message Logs

The message log captures connection information after the first transaction begins and before the next transaction starts.

A connect line that shows the connection handle and database user name is printed in the message log once per database connection.

For example:

```
I. 04/23 13:02:37. 0000000003
Connect:  SA connHandle: 1  SA connID: 3
IQ connID: 0000000003  User: DBA
```

You can use this connection information to match query plans in the `.iqmsg` file with query text. Run the stored procedure **sp_iqcontext** to determine what statement each connection is executing at a given moment.

Insert Notifications in Message Logs

The message log captures notification messages from insert and load operations.

You can turn off notification messages using parameters in the **LOAD** and **INSERT** statements.

See also

• *Columns in the Join Index* on page 225

- *Interpreting Notification Messages* on page 515
- *Load and Insert Messages* on page 259
- *Message Log Wrapping* on page 11

Message Log File Management

By default, the message log file grows to an unlimited size, and exists until you drop the database. To control the size of the message log file, you can set a limit on the size of the file and enable either message log wrapping or log archiving.

You can delete, rename, or copy the message file at any time after stopping the database.

Message log management is controlled by either the server properties **IQMsgMaxSize** and **IQMsgNumFiles**, or the server startup switches **-iqmsgsz** and **-iqmsgnum**:

- **IQMsgMaxSize** or **-iqmsgsz** sets an upper limit in megabytes (MB) on the active message log size. Allowed values are integers between 0 and 2047 (inclusive). The default is 0, which means there is no limit on the size of the message log file.
- **IQMsgNumFiles** or **-iqmsgnum** sets the number of message log archives. Allowed values are integers between 0 and 64 (inclusive). The default is 0, which means that messages are wrapped in the main message log file and there is no archiving.

The value of the server property takes precedence over the corresponding server startup switch. When the server starts, the values of the **-iqmsgsz** and **-iqmsgnum** server switches are written in the server log file.

For information on setting the **IQMsgMaxSize** and the **IQMsgNumFiles** server properties, see *Reference: Building Blocks, Procedures > System Procedures > sa_server_option system procedure*.

For information on setting the **-iqmsgsz** and **-iqmsgnum** server startup switches, see *Utility Guide > start_iq Database Server Startup Utility*.

Note: The **IQMSG_LENGTH_MB** database option has been deprecated; remove it from existing scripts and code. Attempts to use **IQMSG_LENGTH_MB** return an error.

Message Log Wrapping

Message log wrapping overwrites existing messages at the beginning of the file with new messages upon reaching a maximum log size.

When you enable message log wrapping, as soon as the log file reaches the maximum size specified in the **IQMsgMaxSize** server property or the **-iqmsgsz** server startup switch, new messages are written at the beginning of the file. Existing messages are overwritten, line-by-line.

Message log wrapping is enabled by setting **IQMsgMaxSize** or **-iqmsgsz** to a value greater than zero (the default value of zero indicates there is no limit on the size of the message log file) and setting the **IQMsgNumFiles** server property or the **-iqmsgnum** server startup switch to zero (the default).

When wrapping is enabled, the tag `<next msg insertion place>` in the message log file tells you where new messages are being placed. Additional tags at the beginning and end of the file remind you that log wrapping is enabled, and that the last message in the file may not be the most recent one.

See also

- *Columns in the Join Index* on page 225
- *Interpreting Notification Messages* on page 515
- *Load and Insert Messages* on page 259
- *Version Information in Message Logs* on page 9

Message Log Archives

Message log archives limit file size while maintaining all of the information written to the message log file.

A message log archive is a file in which the contents of the active `.iqmsg` message log file is saved.

Enable message log archiving by setting the **IQMsgMaxSize** server property using the **sa_server_option** system stored procedure or the **-iqmsgsz** server startup switch to a value greater than zero, and setting the **IQMsgNumFiles** server property or the **-iqmsgnum** server startup switch to the number of message log archives. The value of the server property takes precedence over the value of its corresponding server switch.

Message Log Archive Names

The names of the `.iqmsg` message log archives follow a pattern.

The pattern for `.iqmsg` message log archive names is `logname.iqmsg.n`, where *n* is an integer greater than zero and `logname` is the name of the message path as specified in the **CREATE DATABASE** statement or is the database name with the suffix `.iqmsg`. The archives are created as read-only files in the same directory as the message log file.

When the message log file `logname.iqmsg` is full and the number of message log archives is fewer than the number specified in **IQMsgNumFiles** or **-iqmsgnum**, the server renames the current message log to create a new archive. A new message log is created with the name `logname.iqmsg`.

For example, if **-iqmsgnum** is equal to 5, the message log archives are created in the following order: `logname.iqmsg.1`, `logname.iqmsg.2`, `logname.iqmsg.3`, `logname.iqmsg.4`, and `logname.iqmsg.5`.

When the message log file `logname.iqmsg` is full and the maximum number of message log archives already exists, the oldest archive (with file extension `.1`) is deleted before the current message log is archived.

For example, when **-iqmsgnum** is 5 and `logname.iqmsg.5` exists:

- The file `logname.iqmsg.1` is deleted.
- The files `logname.iqmsg.2` to `logname.iqmsg.5` are renamed to `logname.iqmsg.1` to `logname.iqmsg.4`, respectively.
- The active message log file is renamed to `logname.iqmsg.5`.
- A new message log file `logname.iqmsg` is created.

Using this method, the server always keeps the most recent message logs, when the value of **IQMsgNumFiles** (or **-iqmsgnum**) is greater than zero.

Message Log Management Errors

Message logging stops if the disk becomes full during message logging. The error `"Disk Full!!! Message logging stopped."` is written in the server log.

Message logging stops if the following errors occur during message log management:

- File rename error: the server cannot archive the current active log or cannot rename any existing log archive
- File creation error: the server cannot create the new message log
- File deletion error: the server cannot delete the old archive
- Disk I/O error

The server automatically resumes message logging when the error condition is resolved. A message is written to the server log when message logging stops and when logging resumes.

For example, if renaming the file fails, the message `"Renaming of <filename> failed. Message log could not be archived. Message logging stopped."` is written in the server log. When logging resumes, the message `"Message logging resumed."` is written in the server log.

If the server fails to delete or rename a file because the file does not exist, the error is ignored and the log management process continues.

When to Back Up Message Log and Archives

Sybase recommends that you create backups for the message log file `dbname.iqmsg` and the message log archives, even though these files are not required for a restore.

If problems occur during a restore, the `.iqmsg` file contains information that proves that the database was shut down before the backup started. The message log files may be useful in diagnosing the cause of the database failure from which you are recovering. Be sure to make a copy before restoring, for use in later analysis.

If IQ message log wrapping is enabled (the **IQMsgMaxSize** server option or the **-iqmsgsz** server startup switch is not equal to zero, and **IQMsgNumFiles** server option or the **-iqmsgnum** server startup switch is zero), you should back up the `.iqmsg` file so that all messages are accessible in the event you need them for diagnostic purposes.

If message log archiving is enabled (the **IQMsgMaxSize** server option or the **-iqmsgsz** server startup switch is not equal to zero and the **IQMsgNumFiles** or the **-iqmsgnum** server startup switch is not equal to zero), the server automatically backs up the message log archives. The maximum amount of message log that is archived is 128GB, which is sufficient in most cases.

Note: Backing up the message log archives *is* required before a server restart. After the server restarts, the existing log archives are ignored and a new archive is created when the `dbname.iqmsg` file is full. To preserve the old archive logs, back up the files before restarting the server.

Daylight Savings Time and Message Logs

If you run Sybase IQ in a country that observes Daylight Savings Time, reboot Sybase IQ servers after the change back to standard time.

The reboot corrects a time difference between the Sybase IQ message log and the SQL log generated by specifying the **-zr** switch on the server startup. Sybase IQ servers also require the reboot after the change to daylight savings time.

The Utility Database

The `utility` database never holds data, but is created automatically for performing certain administrative tasks.

The database server uses the utility database when it needs a database to connect to, but either no real database exists, or none should be running. Sybase IQ installation creates the `utility` database automatically.

Be sure you do not delete this database. You need it to do any of these things:

- Start the database server using the **START ENGINE** command with no database specified
- Create or drop a database when you have no other database to connect to
- Start the database server or connect to a database when any other databases you have are unavailable, for example, due to media failure
- Restore a database

By default, the `utility` database has the user ID `dba` and the password `sql`. You can change these to other values during installation, or later by editing the connection parameters in the `util_db.ini` file in your executable directory.

See also

- *Before You Restore* on page 427
- *Command Line Options that Control Permissions* on page 30
- *Moving Database Files* on page 431
- *Utility Database Server Security* on page 314
- *Verifying a Database Backup* on page 439

Scalability

Sybase IQ's patented design permits databases to scale to contain many terabytes of data.

Its index-based structure allows IQ to store your data in a much smaller space than the size of the raw input data, and access it far faster than a traditional relational database. These features make Sybase IQ ideal for storing and accessing very large databases (VLDBs).

Database administrators need to understand the options and features that affect performance, and follow documented guidelines. While many default settings automatically provide the greatest efficiency, you may need to experiment with certain option settings for the fastest results, based on your configuration, your loading requirements, and your queries. Setting these options appropriately is necessary for top performance in any Sybase IQ database, but is especially important as your database grows to the multiterabyte scale.

This section introduces Sybase IQ features that help you manage a very large database, and points you to more detailed discussion and recommendations.

Memory Use

Allocating memory appropriately is a key factor in performance for all Sybase IQ databases.

Sybase IQ uses memory in its buffer caches for loads and queries. It also uses some memory for managing connections, transactions, buffers, and database objects.

Sybase IQ has two buffer caches, one for the main store and one for the temporary store. The default sizes of these caches are not sufficient for a production data warehouse. You must adjust them to reflect the size of your database and tables, your mix of loads and queries, and other factors such as your operating system and other applications that can affect the amount of memory available.

Data Loads

As your database grows, it is crucial to manage data loading properly.

These features ensure that your loads can scale to meet your needs:

- Buffer manager partitioning to avoid lock contention. Buffer partitioning based on the number of CPUs is enabled by default, and can be adjusted by setting server or database options.
- Allowing sufficient memory for loads, without allocating more memory than is available on your system.
- Reserving space for data structures used during release savepoint, commit, and checkpoint operations.

  See *Reference: Statements and Options > Database Options > General Database Options > MAIN_RESERVED_DBSPACE_MB Option*.

See also

- *Tools for Investigating Lock Contention* on page 380
- *Tune Bulk Loading of Data* on page 297

Processing Threads

Sybase IQ uses operating system threads to process queries and loads.

The default settings of options that control thread use are usually sufficient to provide good performance. In some cases, you may need to change these settings. See *Performance and Tuning Guide > Manage System Resources > The Process Threading Model* to understand how Sybase IQ uses threads. See *Performance and Tuning Guide > Manage System Resources > Options for Tuning Resource Use* to set server options that control thread use.

Disk Space

Learn about managing disk I/O for a Sybase IQ system.

The most important factors in managing disk I/O for a Sybase IQ system are:

- Having enough disk space for queries and loads
- Using that disk space effectively, so that the fastest I/O is available to support the processing speed of high-powered, multi-CPU systems

The **sp_iqstatus** stored procedure indicates the percentage of space used in the IQ main and temporary stores. If there is not enough temporary or main dbspace available for a buffer or dbspace allocation request, then the statement making the request rolls back. You can create a timer-based event to monitor space usage to help avoid unexpected rollbacks, which may occur in out of space situations on non-privileged operations.

Disk striping is an important means of obtaining maximum I/O performance. Disk striping distributes data randomly across multiple disk drives. You can take advantage of disk striping capabilities in your operating system or disk management software and hardware, as well as Sybase IQ internal striping. Sybase IQ disk striping is enabled by default.

See also

- *Data Storage* on page 161
- *Dbspace Management Example* on page 172
- *Locks for DDL Operations* on page 378
- *Versioning of Temporary Tables* on page 375
- *Disconnecting All Connections from a Database in Interactive SQL* on page 159

Intermediate Versioning

A key aspect of managing loads and queries in larger databases is Sybase IQ's transaction-level versioning.

In particular, Sybase IQ offers the ability to roll back transactions to intermediate save points, so that you may not need to repeat the entire load if a long transaction is unable to complete.

See also

- *Transactions and Versioning* on page 365
- *Table Versioning Controls Access to Join Indexes* on page 238

Page Size

When you create your Sybase IQ databases, it is especially important to choose the correct IQ page size.

For very large databases, you need an IQ page size of 128KB or larger.

See also

- *IQ PAGE SIZE Parameter Guidelines* on page 154

Column-based Indexes

Sybase IQ's column-based indexing structure optimizes your ability to perform selections or calculations on attributes of interest to you.

For the best performance, you need the right set of indexes for your data and queries. Your database should have an index on every column that affects performance.

Query Optimizer

The Sybase IQ query optimizer evaluates every query, choosing among various processing options to produce a query plan that offers optimal performance.

The optimizer is tuned for each release of Sybase IQ to choose the best plan for most queries and most databases, including the largest ones.

Schema Design

Sybase IQ often works better with denormalized schemas common in data warehouse design.

In a traditional relational database, normalization improves transaction processing by removing redundancy and improving consistency. In a data warehouse, especially a very large one, denormalization improves performance when processing queries against large amounts of data.

UNION ALL Views

Tables with a large number of rows can have lengthy load times.

The UNION ALL view is one way to address this issue. Sybase IQ lets you partition tables by splitting the data into several separate base tables (for example, by date). You then join them back together into a logical whole by means of a UNION ALL view.

UNION ALL views are simple to administer. If the data is partitioned by, for example, month, you can drop an entire month's worth of data by deleting a table and updating the UNION ALL view definition appropriately. You can have many view definitions for a year, a quarter, and so on, without adding extra date range predicates.

For information on establishing UNION ALL views, and recommendations for optimizing queries that reference these views, see Performance and Tuning Guide > Manage System Resources > Use UNION ALL Views for Faster Loads.

Sybase IQ Startup

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

Ways to Start Database Servers

The first step in running Sybase IQ is to start the database server.

You can start the server in all of these ways:

- Start the server with the Sybase-provided server startup utility, **start_iq**.
- Run the Start Database Server wizard in Sybase Central. See the Online Help for the Sybase Central IQ Plug-in. To start and stop multiplex servers interactively, always use Sybase Central.
- Using the Sybase Control Center. See *Sybase Control Center 3.2.0 > Sybase Control Center for Sybase IQ*.
- Start the server 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. For details, see the *Installation and Configuration Guide*.
- Include a server startline in an ODBC data source.
- Include a server startline in a utility command.
- Issue a SQL command from Interactive SQL to start an additional server.

Note: If you will be using remote data access capabilities to insert data from other databases or to issue queries to other databases, see *System Administration Guide: Volume 2 > Accessing Remote and Data System Administration Guide: Volume 2 > Server Classes for Remote Data Access*.

See also

- *How to Start the iqdemo Database* on page 44
- *International Languages and Character Sets* on page 391
- *Server Names* on page 25
- *Creating an ODBC Data Source from the Command Line* on page 65
- *Starting a Server from Interactive SQL* on page 33
- *ODBC Data Sources* on page 62
- *Ways to Start and Stop Databases* on page 42

- *Connection Parameters in ODBC Data Sources* on page 49
- *JDBC Drivers* on page 51
- *Connecting To the Demo Database From Interactive SQL* on page 54

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 where there are several command line options, we show them for clarity 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. The command line options are one means of tuning Sybase IQ behavior and performance.

See also

- *Command Line Switches* on page 21

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.

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

See also

- *Server and Database Names* on page 24

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 servername.

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 will not start. In the preceding example, the server starts on the default port 2638.

The 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 on failure to start.

All server messages are written to the server log. By default, %IQLOGDIR15% is set by the installation on Windows platforms, and the server log is in %IQLOGDIR15% *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 any of the generic methods to start the database server.
The easiest way to start the server on Windows is from the Start menu. Select Programs > Sybase > Sybase IQ 15.3.

From here, you can start the Sybase IQ Demo database, Sybase Central, 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.

This 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*.

Automatic Server Startup

Use Sybase IQ Service Manager to define a service that will start an IQ server.

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

Command Line Switches

Command line switches are used to define your Sybase IQ environment.

For a complete list of command line switches and full reference information on them, 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. See *Reference: Statements and Options > Database Options and Reference: Statements and Options > SET OPTION Statement*.

See also

- *The Server Startup Utility* on page 20

Displaying Command Line Options

Command line switches are case sensitive.

To display all of the available command line switches, enter the following command at the operating system prompt:

```
start_iq -h
```

Configuration Files

Sets of command line switches can be stored 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 on a server command line. Specify switches in the configuration file as you would on the command line, with the exception that you can enter switches on multiple lines.

Configuration File List

Sybase IQ provides the following configuration files:

Table 2. Configuration Files

File name	Location	Use
`default.cfg`	`$IQDIR15/scripts` (UNIX), `%IQDIR15%\scripts` (Windows)	Generic configuration file. This file is used for default options for **start_iq** and multiplex startup. Sybase IQ copies `default.cfg` into each new database directory and renames it `params.cfg`. Changes to `default.cfg` (in the scripts directory) are inherited by all databases that are created after the file is changed.
`params.cfg`	Database directory	Configuration file created if the database was created by Sybase Central for user-created databases. Changes to this file affect only the server that uses this particular file.

File name	Location	Use
`iqdemo.cfg`	`$IQDIR15/data/demo` (UNIX), `%IQDIR15%\data\demo` (Windows)	Starts the demo database, sets startup switches to the recommended defaults.

Examples of Configuration Files

You can use these 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, with a 20 minute checkpoint interval, allowing anyone to start or stop databases and load data, with user connections limited to 10, a catalog page size of 4096 bytes, default client connection timeout of 72 hours, with TCP/IP as a network protocol and 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 could use these command line options as follows:

```
start_iq @mydb.cfg
```

Note: When you stop the server with the **DBSTOP** command, you need to specify the same parameters as when you started the server. Using a configuration file to start the server ensures that you will be able to find these parameters when you need them.

Command Line Example Formatting in Sybase IQ Documentation

For clarity, examples throughout Sybase IQ documentation show multiple command line switches 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.

Required Command Line Options

Not all command line switches are optional.

While most of the command line switches described in *Utility Guide > Running the Database Server* are optional, you must specify the **-n** switch to run 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, you should include network connection parameters as well. If you start the server without the parameter **-x 'tcpip(port=nnnn)'** set, then 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.

See also

- *Cannot Connect to a Database* on page 492

Default Configuration File

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

This file is used to start servers by Windows services and Sybase Central, and is the source for the `params.cfg` file used by the UNIX **start_iq** command. You can override switches in configuration files by specifying new switches on the **start_iq** command line, except for the *-n servername* switch.

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.

This file can be created when you create the demo database using scripts provided by the 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, Sybase strongly recommends that you run only one database on an IQ server. If you must run multiple databases, start each one on a separate IQ database server, and on a different port.

See also

- *Startup of Database with Database Server* on page 20

Default Server Names

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

Database Names

You can name databases by supplying a **-n** switch following the database file. For example, the following command line starts a database and names it:

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

Naming a database lets you use a nickname in place of a file name that may be difficult to remember.

Server Names

You name the 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, the following command line 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. It is especially important in a multiuser, networked environment where shared memory will be used for local database connections. This convention ensures that all users will be able to connect to the correct database, even when other databases with the same name have been started on other host systems.

To allow Sybase IQ to locate the server no matter what character set is 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. The following 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 better 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 in order to start the server without starting a specific database.

See also

- *How to Start the iqdemo Database* on page 44
- *International Languages and Character Sets* on page 391
- *Ways to Start Database Servers* on page 19
- *Creating an ODBC Data Source from the Command Line* on page 65
- *Starting a Server from Interactive SQL* on page 33

Case Sensitivity and Naming Conventions

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

You should 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.

Command-line Options for Performance

Several command line options can affect database server performance.

Most of the options described in this section 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. If you need to specify options that affect the catalog store only, see *Reference: Statements and Options > Database Options*.

Performance tuning suggestions are given throughout this guide.

Memory Options

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 for these purposes.

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.

You need to set the buffer cache sizes for the IQ main and temporary stores in one of two ways:

- 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). Recommended method. For syntax, see the *Utility Guide > start_iq Database Server Startup Utility*.
- 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 will accommodate, however, 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 you should use **-iqmc** and **-iqtc**. The **-iqmc** and **-iqtc** options do not override settings by **sa_server_option** procedure.

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.

Buffer partitioning based on the number of CPUs is enabled by default, and can be adjusted by setting the **-iqpartition** option of the `start_iq` database startup utility:

- **-iqpartition –** Specifies the number of partitions in the IQ main and temp buffer caches. Must be a power of 2. Allowed values are: 0 (default), 1, 2, 4, 8, 16, 32, 64. By default, 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 IQ server, and overrides the value set by the `Cache_Partitions` database option.

For syntax, see *Utility Guide > start_iq Database Server Startup Utility* and *Reference: Statements and Options > Database Options > Alphabetical List of Options > CACHE_PARTITIONS Option*.

Concurrent User Switches

Your license sets the absolute number of concurrent users.

You must also set the **-gm** switch. This required switch lets you limit the number of concurrent user connections on a particular server.

The **-gn** switch sets the number of execution threads that will be 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 the following formula:

```
gn_value = gm_value + 5
```

Specify a minimum of 25.

On UNIX platforms, see the *Installation and Configuration Guide* for your platform for more information.

There may be times when you want to tune performance for a particular operation by limiting the number of user connections to fewer than your license allows. Alternatively, you may want to use the **-iqgovern** switch to control query use.

Concurrent Query Switch

The **-iqgovern** switch lets you specify the number of concurrent queries on a particular server.

This is not the same as the number of connections, which is controlled by your license. By specifying the **-iqgovern** switch, you can 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, try setting **-iqgovern** slightly lower.

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 can cause serious performance degradation.

Number of Processing Threads

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

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 can improve performance.

Number of Processors

If you are running on a multiprocessor machine, you can set the number of processors used by the database server for catalog store operations with the **-gt** option.

By default, all available processors are used.

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. Sybase recommends that you use one of these methods.

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.

Any cache size less 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.

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
- 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.

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

For more information on setting the catalog cache size, see the *Utility Guide*.

Number of CPUs Switch

The **-iqnumbercpus** switch on the Sybase IQ startup command specifies the number of CPUs available to IQ.

It overrides 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: Sybase recommends using **-iqnumbercpus** only in the following situations:

- On machines with Intel® CPUs and hyperthreading enabled, setting **-iqnumbercpus** to the number of CPUs available
- On machines where an operating system utility has been used to restrict Sybase IQ to a subset of the CPUs within the machine

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

Options That Affect Timing

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

Checkpoint Interval

Sybase IQ uses checkpoints to generate reference points and other information that it needs to recover databases.

Use the **-gc** switch to set the maximum number of minutes the database server will run without doing 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 this 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 will take to recover from system failure.

When a database server is running with multiple databases, the recovery time specified by the first database started will be used unless overridden by this switch.

Network Performance Options

Several options help you tune network performance.

They include **-gb** (database process priority on Windows), and **-p** (maximum packet size).

Command Line Options that Control Permissions

Some command line options control the permissions required to carry out certain global operations.

Switches that Start and Stop Databases

The **-gd** option allows you to limit the users who can start or stop a database on a running server to those with a certain level of permission in the database to which they are already connected:

- **DBA** — Only users with DBA authority 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. (Note that users still need permission to access a particular database once they have 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 the DBA 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 the DBA. However, it also lets non-DBAs stop a database. For this reason, some sites 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 permission in the database to which they are connected.

- **DBA**—Only users with DBA authority 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.

The 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 DBA authority 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 permission in the database.

- **DBA** — Only users with DBA authority can load data.
- **ALL** (default for **start_iq** and `default.cfg`) — Any user can load data.
- **NONE** — Data cannot be loaded.

See also

- *Before You Restore* on page 427
- *Moving Database Files* on page 431
- *The Utility Database* on page 14
- *Utility Database Server Security* on page 314
- *Verifying a Database Backup* on page 439

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.

You 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 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, remember to increase your cache size. A cache of the same size will accommodate 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 to the catalog store only. You set the page size for the IQ store in the **IQ PAGE SIZE** parameter of the **CREATE DATABASE** command.

See also

- *IQ PAGE SIZE Parameter Guidelines* on page 154

Client/Server Environment Options

Three options can help you set up your client/server environment.

- **-x** specifies communication protocol options.
- **-tl** sets the network connection timeout.

- **-ti** sets the client connection timeout.

Communications Protocols

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

The database server supports the following 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. At the client side, many of the same options can be controlled using the **CommLinks** connection parameter.

The following command starts a server using the TCP/IP protocol:

```
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
```

See also

- *Network Communications Parameters* on page 109

Switches that Limit Inactive Connections

Sybase IQ uses two switches, **-tl** and **-ti**, to determine when it should close user connections.

Default Network Timeout Switch

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 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 Sybase IQ, will 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** discussed in the next section. 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

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 will not be rolled back.

For more information, see *Utility Guide > start_iq Database Server Startup Utility > start_iq Server Options*.

Forced Recovery Mode Options

Should you need to restart your server after a failure, you can usually do so using the same startup options as usual.

On rare occasions, you may need to supply startup options to force recovery or to recover leaked storage. For server options, see *Utility Guide > start_iq Database Server Utility > start_iq Server Options*.

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.

See also

- *Creating an ODBC Data Source from the Command Line* on page 65
- *How to Start the iqdemo Database* on page 44
- *International Languages and Character Sets* on page 391
- *Server Names* on page 25
- *Ways to Start Database Servers* on page 19

Shared Memory Conflicts

In a production environment, it would be unusual to have more than one server running on the same system, and Sybase strongly recommends against doing this. 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 the following message:

```
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 is unable to create a shared memory segment. This error may occur when either a 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 Sybase IQ or SQL Anywhere.

Ways to Avoid Shared Memory Conflicts

To avoid conflicts when using shared memory, consider doing one or more of the following:

- Create a temporary directory dedicated to each server. Make sure that each client uses the same temporary directory as its server by setting the IQTMP15 environment variable explicitly on both systems. For details about setting environment variables, see *Reference: Building Blocks, Tables, and Procedures > File Locations and Installation Settings > Environment Variables*.
- For each server, create a data source 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 the following command:

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

If you run multiple servers per system, Sybase IQ requires that you:

- Make sure that each server has a unique name, specified with the **-n** parameter on startup.
- Make sure that each server has a unique port number, specified with the **-x** parameter.

For examples using these parameters, see the *Utility Guide > start_iq Database Server Startup Utility*.

See also

- *ODBC Data Sources on UNIX* on page 68

Server Activity Logs

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

Server Startup Messages

When you start an 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. The following is an example of what you see on AIX:

```
Starting server myserver_iqdemo on myserver at port 2638 (05/22
16:18:58)
Run Directory       : /myserver/users/sybase/iq153/IQ-15_3/demo
Server Executable   : /myserver/users/sybase/iq153/IQ-15_3/bin64/
iqsrv15
Server Output Log   : /myserver/users/sybase/iq153/IQ
15_3/logfiles/myserver_iqdemo.0001.srvlog
Server Version      : 15.3.0.5027/GA
Open Client Version : 15.0/P-EBF16070 ESD #15
User Parameters     : '@iqdemo.cfg' 'iqdemo.db'
Default Parameters  : -ti 4400 -gn 25

I. 05/22 16:19:05.      Sybase IQ
I. 05/22 16:19:05.       Version 15.3
I. 05/22 16:19:05.        (64bit mode)
I. 05/22 16:19:05. Copyright 1992-2010 by Sybase, Inc. All rights
reserved
I. 05/22 16:19:05.
I. 05/22 16:19:05. 4 physical processor(s) detected.
I. 05/22 16:19:05. Maximum number of physical processors the
server will use: 4
I. 05/22 16:19:05. Running AIX 5 3 on PPC
I. 05/22 16:19:05. Server built for PPC processor architecture
I. 05/22 16:19:05. 49152K of memory used for caching
```

```
I. 05/22 16:19:05. Minimum cache size: 49152K, maximum cache size:
262144K
I. 05/22 16:19:05. Using a maximum page size of 4096 bytes
I. 05/22 16:19:05. Starting database "iqdemo"
(/myserver/users/sybase/iq151/IQ-15_3/demo/iqdemo.db)
at Fri May 22 2009 16:19
=============================================================
IQ server starting with:
    10 connections         (       -gm )
    18 cmd resources       ( -iqgovern )
    267 threads             (     -iqmt )
    512 Kb thread stack size   (   -iqtss  )
    136704 Kb thread memory size ( -iqmt * -iqtss )
    4 IQ number of cpus  ( -iqnumbercpus )
    0 MB maximum size of IQMSG file ( -iqmsgsz )
    0 copies of IQMSG file archives ( -
iqmsgnum )===========================================================
====
I. 05/22 16:19:07. Transaction log: iqdemo.log
I. 05/22 16:19:08. Starting checkpoint of "iqdemo" (iqdemo.db) at Fri
May 22 2009 16:19
I. 05/22 16:19:08. Finished checkpoint of "iqdemo" (iqdemo.db) at Fri
May
22 2009 16:19
=============================================================
IQ server starting with:
    10 connections         (       -gm )
    18 cmd resources       ( -iqgovern )
    267 threads             (     -iqmt )
    512 Kb thread stack size   (   -iqtss  )
    136704 Kb thread memory size ( -iqmt * -iqtss )
    4 IQ number of cpus  ( -iqnumbercpus )
    0 MB maximum size of IQMSG file ( -iqmsgsz )
0 copies of IQMSG file archives ( -iqmsgnum )
=============================================================
I. 05/22 16:19:07. Transaction log: iqdemo.log
I. 05/22 16:19:08. Starting checkpoint of "iqdemo"
(iqdemo.db)
at Fri May 22 2009 16:19
I. 05/22 16:19:08. Finished checkpoint of "iqdemo" (iqdemo.db)
at Fri May 22 2009 16:19I. 05/22 16:19:10.
Database "iqdemo" (iqdemo.db) started at Fri May 22 2009 16:19
I. 05/22 16:19:10. IQ Server myserver_iqdemo.
I. 05/22 16:19:10. Database server started at Fri May 22 2009 16:19
I. 05/22 16:19:10. Trying to start SharedMemory link ...
I. 05/22 16:19:10.     SharedMemory link started successfully
I. 05/22 16:19:10. Trying to start TCPIP link ...
I. 05/22 16:19:15.     TCPIP link started successfully
I. 05/22 16:19:15. Now accepting requests
```

```
New process id is 397436
Server started successfully
```

```
I. 05/20 22:11:26. Starting database "iqdemo"
(/testproduction/users/miller/setup_main/iqdemo.db)
at Wed May 20 2009 22:11
```

```
I. 05/20 22:11:27. Transaction log: iqdemo.log
I. 05/20 22:11:27. Starting checkpoint of "iqdemo" (iqdemo.db)
at Wed May 20 2009 22:11
I. 05/20 22:11:27. Finished checkpoint of "iqdemo" (iqdemo.db)
at Wed May 20 2009 22:11
I. 05/20 22:11:28. Database "iqdemo" (iqdemo.db) started
at Wed May 20 2009 22:11
I. 05/20 22:11:28. IQ Server iqdemo1.
I. 05/20 22:11:28. Database server started
at Wed May 20 2009 22:11
I. 05/20 22:11:28. Trying to start SharedMemory link ...
I. 05/20 22:11:28.     SharedMemory link started successfully
I. 05/20 22:11:28. Trying to start TCPIP link ...
I. 05/20 22:11:28. Starting on port 2638
I. 05/20 22:11:33.     TCPIP link started successfully
I. 05/20 22:11:33. Now accepting requests
I. 05/20 22:11:33. Press 'q' to shut down the database server
I. 05/20 22:11:35. Database server shutdown requested by pressing Q
I. 05/20 22:11:35. Starting checkpoint of "iqdemo" (iqdemo.db)
at Wed May 20 2009 22:11
I. 05/20 22:11:35. Finished checkpoint of "iqdemo" (iqdemo.db)
at Wed May 20 2009 22:11
I. 05/20 22:11:35. Database server stopped at Wed May 20 2009 22:11
```

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 $IQLOGDIR15.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 don’t define $IQLOGDIR15 directory, then on UNIX, the log is written to `$IQDIR15/logfiles/` directory, and on Windows to the*$IQLOGDIR15* directory defined by the Sybase IQ installation.

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

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

- Log into the machine where the server was started, and issue the command:

  ```
  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/iq152/IQ-15_3/bin64/
  iqsrv15
  @iqdemo.cfg iqdemo.db -ti 4400 -gn 25 -o
  /ibm64srv/users/sybase/iq152/IQ-
  15_3/logfiles/ibm64srv_iqdemo.0003.srvlog -hn 7
  ```
- Use the **stop_iq** utility, described in the following section, which displays all Sybase IQ processes running.

On Windows systems, look in the system tray for one or more 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** startup 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 as follows:

```
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, 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 need to know not only how to stop the IQ database server, but also 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 a limited number of situations the server may need to be stopped and restarted.

These situations include:

- To install a new version of Sybase IQ
- To reset some server command line options
- To cause a small number of server-wide database options to take effect. See *Reference: Statements and Options > Database Options > Introduction to Database Options > Scope and Duration of Database Options* for a complete list.
- Before closing the operating system session

Ways to Stop Database Servers

The preferred methods of stopping the database server in Sybase IQ are as follows:

- In Sybase Central (either UNIX or Windows), right-click the server name and choose Stop Server.

  To shut down servers in an IQ multiplex, open the Multiplex folder, right-click the server that needs to be stopped and choose **Control > Stop**.
- In UNIX, use the **stop_iq** utility at the operating system command line. For details, see *Utility Guide > stop_iq Database Shutdown Utility*.

  When you run **stop_iq**, it displays the following message:

  ```
  "Please note that 'stop_iq' will shut down a server
  completely without regard for users, connections, or load
  process status. For more control, use the 'dbstop' utility,
  which has options that control stopping servers based on
  active connections."
  ```
- In Windows, click **Shutdown** on the database server display or right-click the IQ icon in the system tray and select **Exit**.
- In Windows, if the server is run as a service, open the Service Manager in Control Panel. Select the service and click **Stop**.

Normally, you should not shut down a server while it is still connected to one or more clients. If you try this, you get a warning that any uncommitted transactions will be lost. Disconnect or close all the clients and try again.

You can also stop the database server in the following ways:

- At the operating system command line, issue the **DBSTOP** command with appropriate parameters. Use the same parameters as when you started the server. Without the proper connection parameters **DBSTOP** does not know how to connect to the server to tell it to shut down. For details on using **DBSTOP**, see *Utility Guide > dbstop Database Shutdown Utility*.
- In an Interactive SQL window or command file, issue the **STOP ENGINE** command to stop a named database server.

- In UNIX, in the window where the database server was started, type:

  ```
  q
  ```

  This command does not work if you have redirected input to a different device or if you started the server with **start_iq**. It only works with **iqsrv15**.
- In a UNIX **cron** or **at** job, use **stop_iq** with the appropriate **-stop** option. The utility stops one or all servers associated with the user who starts the **cron** or **at** job depending on the parameter specified. The user must be the same one who started the server. No operator prompting occurs, and no operator action is required.
  To use **stop_iq** in such jobs, specify the utility with the appropriate **-stop** option:

  ```
  stop_iq -stop one
  ```

  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.

  ```
  stop_iq -stop all
  ```

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

  Note: You must specify the full path name to the **stop_iq** executable in the **cron** statement.

Example — Stop a Server With stop_iq

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

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

```
% stop_iq
```

```
Checking system for IQ 15 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-15_3/logfiles/surya_ibm2.001.srvlog -hn 8      pamela    409802
18:05:02    0:05  SVR:ibm1_iqdemo2 DB:iqdemo
PORT:2678/ibm1/users/sybase/iq151/IQ-15_3/bin64/iqsrv15 @iqdemo.cfg
iqdemo.db -ti 4400 -gn 25 -o /ibm1/users/sybase/iq151/IQ
15_3/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/iq151/IQ-15_3/bin64/
iqsrv15
@iqdemo.cfg iqdemo.db -ti 4400 -gn 25 -o /myserver/users/sybase/
iq151/IQ-
15_3/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). Messages like the following display:

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

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

If no running servers were started by your user ID, Sybase IQ displays information about servers run by other users, then a message like the following:

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

Example —Stop 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.

Who Can 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**, but you can also set the value to **ALL** or **NONE**. If you set it to **NONE**, even the DBA cannot execute **STOP ENGINE**. In a production environment, Sybase strongly recommends that only the DBA 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 to stop the database server (other than the UNIX command shown in the previous section), 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.

See also

- *Data Protection* on page 405
- *Resource Issues Running sp_iqcheckdb* on page 461
- *System Recovery* on page 385
- *System Recovery and Database Repair* on page 453
- *Validating Your Database* on page 426

Ways to Start and Stop Databases

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

Sybase recommends that you run only one database per server, especially in a production environment.

Ways to Start a Database on a Running Server

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. See *Reference: Statements and Options > SQL Statements > START DATABASE Statement*.
- To start and connect to a database from Interactive SQL or Sybase Central, 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 parameter **"DBF=db-file"** in the connection parameters.
- To start a database from Sybase Central, see *Introduction to Sybase IQ*.
- To start an embedded database, while connected to a server, connect to a database using a DBF parameter. This parameter specifies a database file for a new connection. The database file is loaded onto the current server.

See also

- *ODBC Data Sources* on page 62

- *Ways to Start Database Servers* on page 19
- *Connection Parameters in ODBC Data Sources* on page 49
- *JDBC Drivers* on page 51
- *Connecting To the Demo Database From Interactive SQL* on page 54

Database Startup Guidelines

Several issues affect database startup.

File Access

In order 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. A database can be started 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, 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 all files of IQ_SYSTEM_TEMP cannot be opened, the database will not start unless the **-iqnotemp** startup parameter is used.

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. Note that 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 pages of a fixed size. Once a server has been started, you cannot load a database that has a larger catalog page size or IQ page size than the server. For this reason, you should 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 database administrator privileges 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 DBA, can start a database.

See also

- *Catalog Store* on page 136

Ways to Stop Databases

You can stop a database in the following ways:

- 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 *Reference: Statements and Options > SQL Statements > STOP DATABASE Statement [Interactive SQL]*.

How to Start the iqdemo Database

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

This configuration file, called `iqdemo.cfg`, contains all the parameters necessary to start the demo database. See *Quick Start > Demo database > Creating and using an IQ demo database*.

See also

- *International Languages and Character Sets* on page 391
- *Server Names* on page 25
- *Ways to Start Database Servers* on page 19
- *Creating an ODBC Data Source from the Command Line* on page 65
- *Starting a Server from Interactive SQL* on page 33

Sybase Central Start and Stop Methods

If your system supports a graphical user interface, you may use Sybase Central to perform many administrative tasks.

See *Introduction to Sybase IQ > Running and Connecting to Servers* or use the online help.

Sybase IQ Connections

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. The connection options you choose must take these factors into account.

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

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

For more information on connecting to a database from Sybase Open Client™ applications, see *System Administration Guide: Volume 2: > Sybase IQ as a Data Server*.

For more information on connecting via JDBC (if you are not working in Sybase Central or Interactive SQL), see *SQL Anywhere Server – Programming*.

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.

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 ])
```

How Sybase IQ Establishes Connections

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

Sybase IQ supports the following interfaces:

- ODBC — ODBC connections are discussed in this chapter.
- OLE DB — OLE DB connections are discussed in this chapter.
- Embedded SQL — Embedded SQL connections are discussed in this chapter.
- — This chapter does not describe Open Client connections. For information on connecting to IQ from Open Client applications, see *System Administration Guide: Volume 2 > Sybase IQ as a Data Server*.
- JDBC — This chapter describes JDBC connections. For more information on connecting via JDBC, see *Introduction to Sybase IQ > Managing Databases*. To create JDBC data sources, see *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL Anywhere JDBC driver > Using the iAnywhere JDBC driver*.

Note: JDBC provides the link between the execution of Java objects and database operations.

The interface uses connection information included in the call from the client application, possibly together with connection information stored on disk in a file data source, to locate and connect to a server running the required database.

See also

- *Java Support Guidelines* on page 156
- *Default Connection Parameters* on page 61
- *Connecting From Sybase IQ Utilities* on page 62

Learning Roadmap for Connections

Connection topics are designed for a variety of IQ users. The roadmap shows which topics address specific connection needs.

If you want ...	Click the related topic link at the bottom of this page ...
An overview of connecting from Sybase Central or Interactive SQL (including a description of the drivers involved)	*Sybase Central or Interactive SQL Connections*
Some examples to get started quickly	*Simple Connection Examples*

If you want ...	Click the related topic link at the bottom of this page ...
A conceptual overview	*How Connection Parameters Work*
To learn what connection parameters are available	*Connection and Communication Parameters*
To create data sources	*ODBC Data Sources*
To see an in-depth description of how connections are established	*ODBC Data Sources*
To add users and grant them permissions	*Database Permissions and Authorities Overview*
To diagnose network-specific connection issues	*Troubleshooting Network Communications*
To learn about character set issues affecting connections	*Connection Strings and Character Sets*

See also

- *Connection and Communication Parameters* on page 83
- *Database Permissions and Authorities Overview* on page 303
- *How Connection Parameters Work* on page 47
- *ODBC Data Sources* on page 62
- *Simple Connection Examples* on page 53
- *Sybase Central or Interactive SQL Connections* on page 49
- *Troubleshooting Network Communications* on page 509

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

- *Learning Roadmap for Connections* on page 46
- *CharSet connection parameter [CS]* on page 88
- *Language Connection Parameter [LANG]* on page 102
- *Setting a Locale* on page 400

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

- 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.
- 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.

See also

- *ODBC Data Sources* on page 62
- *Ways to Start and Stop Databases* on page 42
- *Ways to Start Database Servers* on page 19
- *JDBC Drivers* on page 51
- *Connecting To the Demo Database From Interactive SQL* on page 54

Sybase Central or Interactive SQL Connections

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

In the Connect dialog, you tell Sybase Central or 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. Sybase Central or 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 Sybase Central or Interactive SQL connects to the right database.

This section describes how to access the Connect dialog in Sybase Central and 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

- *Learning Roadmap for Connections* on page 46
- *Connection and Communication Parameters* on page 83
- *Simple Connection Examples* on page 53

The Connect Dialog

The Connect dialog lets you define parameters for connecting to a server or database. The same dialog is used in both Sybase Central and Interactive SQL.

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 tab lets you add connection parameters and specify a driver for the connection.

After you connect successfully, the database name appears in the left pane of the main window, under the server that it is running on. The user ID for the connection is shown 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 (Sybase Central)

A Connect dialog in Sybase Central lets you connect to a database.
In Sybase Central, choose Connections > Connect with Sybase IQ 15...

If you have more than one Sybase Central plug-in installed, choose Sybase IQ from the list.

You can also click the Connect button on the main toolbar or press F11 to open the Connect dialog.

Note: You can make subsequent connections to a given database easier and faster using a *connection profile*.

See also

- *Connecting To a Running Database On a Local Server* on page 55
- *Connecting to a database that is not yet running* on page 56
- *Simple Connection Examples* on page 53

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 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.

See also

- *Creating a File Data Source Using the ODBC Administrator* on page 68
- *Connecting Using a Data Source* on page 60

JDBC Drivers

Interactive SQL and Sybase Central support two main JDBC drivers: Sybase jConnect™, and the iAnywhere JDBC Driver. Both are included with Sybase IQ.

By default, the Sybase Central IQ plug-in and Interactive SQL use the iAnywhere JDBC Driver. It provides JDBC 2.0 support and fully scrollable cursors, which the jConnect 5.5 driver does not. jConnect is still useful for zero footprint applications such as Web pages.

For more information on JDBC drivers, including required software, see *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL*

Anywhere JDBC driver > Introduction to JDBC > Choosing a JDBC driver and *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL Anywhere JDBC driver > Using the jConnect JDBC driver* .

See also

- *ODBC Data Sources* on page 62
- *Ways to Start and Stop Databases* on page 42
- *Ways to Start Database Servers* on page 19
- *Connection Parameters in ODBC Data Sources* on page 49
- *Connecting To the Demo Database From Interactive SQL* on page 54

Connection Shortcuts in Sybase Central

By following the procedures in these topics, you can simplify Sybase Central connections.

Server Objects

To insert from an Adaptive Server Enterprise database to a Sybase IQ database, each server must have an entry, also called a *server object*, in the interfaces file.

Sybase IQ provides a shortcut to connection information that you can use to:

- Connect using IQISQL
- Simplify database startup
- Simplify connection from Sybase Central

Use IQDSEDIT (Directory Services Editor) to create entries in the interfaces file. You must be the owner of the Sybase home directory (%SYBASE%) or have write permission in order to run IQDSEDIT.

Once you add servers to this file, the Server Name dropdown box is enabled wherever Sybase Central requests connection information. When you tab to the dropdown box, pressing the space bar lists all the entries you created with IQDSEDIT. You can choose a server from the list or just press the first letter of the server name you want to use. Pressing the same letter multiple times cycles through all the values that begin with that letter.

Creating a Connection Profile

Connection profiles make it easy to connect to databases automatically when an individual user boots a system, or to connect without typing connection parameters.

1. From the Sybase Central menu, choose Connections > Connection Profiles or F9.
2. On the Connection Profiles dialog, click New, and create the profile.
3. Click OK.

Because connection profiles include the username and password, they are better suited to individual use than across an entire installation. Sybase Central lets you choose from a list of existing profiles, create, edit, or delete a profile, or select a profile to be used automatically when you start Sybase Central.

If you would like Sybase Central to connect to this connection each time you start your computer, click Set Startup (Alt-S) on the Connection Profiles window.

If you choose not to connect automatically on startup, you can now connect from Sybase Central by simply choosing Connections > Connection Profiles and clicking Connect.

Simple Connection Examples

Although the connection model for Sybase IQ is configurable, and can become complex, in many cases connecting to a database is very simple.

This section describes some simple cases of applications connecting to a Sybase IQ database. When you are getting started, for example, if you are running the `iqdemo` database on a local server and are not connected to a network, you may only need the instructions in this section. However, in most IQ environments, to ensure that you can connect and disconnect properly, a complete set of connection parameters is essential.

For steps in connecting to a database using Sybase Central, see the *Introduction to Sybase IQ*.

See also

- *Learning Roadmap for Connections* on page 46
- *Connection and Communication Parameters* on page 83
- *Sybase Central or Interactive SQL Connections* on page 49
- *Connecting To a Running Database On a Local Server* on page 55
- *Opening the Connect dialog (Sybase Central)* on page 50
- *Connecting to a database that is not yet running* on page 56
- *Connection and Communication Parameters* on page 83
- *Connection Strings and Character Sets* on page 398

Connecting to the demo database from Sybase Central

Many examples and exercises throughout the documentation start by connecting to the demo database.

1. Start Sybase Central as appropriate for your system.

   On UNIX, source the `IQ-15_3.csh` (or `.sh`) script before invoking utilities like Sybase Central or the IQ Agent.

   In a multiplex environment, if the IQ Agent is not started, type:

   ```
   %S99SybaseIQAgent15
   ```

To start Sybase Central, type:

```
% scjview
```

Note: If you have set environment variables as described in the Installation and Configuration Guide, you can issue the `scjview` command from any directory.

On Windows, to start Sybase Central, choose Programs > Sybase > Sybase IQ 15.3 > Sybase Central.

2. Choose Sybase IQ.

   This opens a panel on the right with multiple tabs.
3. In the Utilities tab, double-click Open Interactive SQL.
4. On the Identification tab, type `DBA` and `sql` for the User and Password.
5. On the Database tab, choose Find.
6. Select your iqdemo server from the Find Servers screen and click OK.

You can connect to any database server that is already running in the same manner. You can also specify a non-default character set and language.

See also

- *Additional Dbspaces* on page 169
- *Creating Primary Keys* on page 186

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 15.3 > Interactive SQL**.
2. Follow steps 4 – 6 in the previous procedure.

You can connect to any database server that is already running in the same manner. You can also specify a non-default character set and language.

For more information on using **Interactive SQL**, see *Utility Guide > dbisql Interactive SQL Utility*.

See also

- *ODBC Data Sources* on page 62
- *Ways to Start and Stop Databases* on page 42
- *Ways to Start Database Servers* on page 19
- *Connection Parameters in ODBC Data Sources* on page 49
- *JDBC Drivers* on page 51
- *Additional Dbspaces* on page 169

- *Creating Primary Keys* on page 186

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, you need to identify the database file so that Sybase Central or Interactive SQL can start it for you.
- Are there multiple databases running on your machine? If so, you need to identify the database to which you want Sybase Central or Interactive SQL to connect. If there is only one database, Sybase Central or Interactive SQL assumes that you want to connect to that one, and you don't 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 Sybase Central or Interactive SQL and open the Connect dialog (if it doesn't appear automatically.)
2. On the Identification tab of the dialog, enter a user ID and a password.
3.

   Do one of the following:
   - If the server only contains the one database, click OK to connect to it.
   - If the server contains multiple databases, click the Database tab of the dialog 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 only need to provide a database name for a successful connection. The database file is not necessary.

See also

- *Opening the Connect dialog (Sybase Central)* on page 50
- *Connecting to a database that is not yet running* on page 56
- *Simple Connection Examples* on page 53

Connecting to a database that is not yet running

If the database is not yet running, you must specify the database file name and path to connect.

1. Start Sybase Central or Interactive SQL and open the Connect dialog (if it doesn't appear automatically).
2. Open the Identification tab of the dialog, enter a user ID and a password.
3. Click the Database tab of the dialog.
4. Specify a file in the Database File field (including the full path, name, and extension). You can search for a file by clicking Browse.
5. If you want the database name for subsequent connections to be different from the file name, enter a name in the Database Name field (without including a path or extension).

See also

- *Connecting To a Running Database On a Local Server* on page 55
- *Opening the Connect dialog (Sybase Central)* on page 50
- *Simple Connection Examples* on page 53

Connecting to a database from Interactive SQL on UNIX

The following procedure shows how to connect to a running database from Interactive SQL.

1.

   Start the server and the database by typing at a system command prompt:

   ```
   start_iq dbname
   ```

2.

   Start Interactive SQL by typing at a system command prompt:

   ```
   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`, you enter:

   ```
   dbisql -c "uid=DBA;pwd=sql" -host fiona -port 1870 -n fiona_iqdemo
   $IQDIR15/demo/iqdemo.db
   ```

   You do not need to specify the host and port if you are connecting to a database on your local machine.

Connecting from a UNIX system

The following procedure shows how to connect to a running database from the command line on a UNIX system.

1. Make sure that your PATH and other environment variables are correctly set, as described in *Reference: Building Blocks, Tables, and Procedures > File Locations and Installation Settings*.
2. To ensure that the demo database is loaded on a running server, at the UNIX prompt, enter:

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

   To start the demo database, enter:

   ```
   cd $IQDIR15/demo
   ```

   ```
   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**. Note that although **dbisqlc** is supported, **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(host=SERV2;port=1234)"
   ```

   If you prefer, use this alternate form of the links clause, which has the same result:

   ```
   "links=tcpip(host=SERV2:1234)"
   ```

Connecting From a Windows System

The following procedure shows how to connect to a running database from the command prompt on a Windows system.

1. Choose **Programs > Sybase > Sybase IQ 15.3 > Interactive SQL**, or at the Windows command prompt, enter:

   ```
   dbisql
   ```

   You can include the `-c` parameter to specify connection parameters in the **dbisql** command, as described in the procedure above for connecting to UNIX. 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 you enter `DBA` and `sql`, the default user and password combination for 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 15.3 > ODBC Administrator 32-bit** or **Sybase IQ 15.3 > ODBC Administrator 64-bit**. Select the **User Data Sources on the User DSN** tab, then click **Configure**. You can find the port number by typing `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 the Connect dialog or an error message about missing information pops up, you may need to 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, as in:

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

   If you connect using Interactive SQL Classic, the words "Connected to database" appear in the Statistics window along with a message displaying 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. In this case, 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.

The following connection parameters show how to load the demo database as an embedded database:

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

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

Start Parameter

The following connection parameters show how you can customize the startup of the demo database as an embedded database. This is useful if you wish to 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 Interactive SQL with no databases running. You can use either of the following ways:
   - From the Windows Programs menu, choose Sybase > Sybase IQ 15.3 > 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-15_3`, enter:

   ```
   c:\sybase\IQ-15_3\iqdemo.db
   ```
5. Leave all other fields blank, and click OK. Sybase IQ starts up and loads the demo database, and Interactive SQL connects to the database.

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.

See also

- *Creating a File Data Source Using the ODBC Administrator* on page 68
- *Drivers for Connections* on page 51

Connecting from Sybase Central or Interactive SQL using a Data Source

Connect from Sybase Central or Interactive SQL using an ODBC data source.

1. Start Sybase Central or 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 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 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
```

See also

- *How Sybase IQ Establishes Connections* on page 46
- *Java Support Guidelines* on page 156
- *Connecting From Sybase IQ Utilities* on page 62

Connecting From Sybase IQ Utilities

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:

1. 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{host=localh
   ost:2638}' -d  '/mydir'
   ```
2. Using the SQLCONNECT environment variable settings if any command line values are missing. 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*.
3. 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*.

See also

- *How Sybase IQ Establishes Connections* on page 46
- *Java Support Guidelines* on page 156
- *Default Connection Parameters* on page 61

ODBC Data Sources

You can store a set of 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.

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

- *Ways to Start and Stop Databases* on page 42
- *Ways to Start Database Servers* on page 19
- *Connection Parameters in ODBC Data Sources* on page 49
- *JDBC Drivers* on page 51
- *Connecting To the Demo Database From Interactive SQL* on page 54
- *Learning Roadmap for Connections* on page 46

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 15.3 > ODBC Administrator 32-bit

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

   The Create New Data Source wizard appears.
3. Select the 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 on system PUSHKIN using TCP/IP protocol and port 1870, you would click **TCP/IP** and type:

   ```
   host=pushkin:1870
   ```

   You could also use the host network address. For example:

   ```
   host=157.133.66.75:1870
   ```
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 the connection parameters you wish to use. For example, the following command creates a data source for the 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-15_3\demo
   \iqdemo.db"
   ```

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

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

The **iqdsn** utility lists 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*.

See also

- *Starting a Server from Interactive SQL* on page 33
- *How to Start the iqdemo Database* on page 44
- *International Languages and Character Sets* on page 391
- *Server Names* on page 25
- *Ways to Start Database Servers* on page 19

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 15.3 > 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 Sybase IQ connection parameters.

See also

- *Creating a File Data Source Using the ODBC Administrator* on page 68

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 a 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).

Area	Description
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 Sybase IQ 15 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 Sybase IQ 15 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.

See also

- *DataSourceName Connection Parameter [DSN]* on page 95
- *FileDataSourceName Connection Parameter [FileDSN]* on page 99

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.

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 Sybase IQ 12 from the list of drivers, and click Next.
3. Follow the instructions to create the data source.

See also

- *Connecting Using a Data Source* on page 60
- *Drivers for Connections* on page 51
- *Configuring ODBC Data Sources in ODBC Administrator* on page 66

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*.

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, you must use the long form of each identifier, for example:

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

In addition to the connection parameters in the sample DSN above, you can include others in the .odbc.ini file as well. See *Connection Parameters* for information about all supported connection parameters. Network communication parameters are added as part of the

CommLinks (LINKS) parameter. For a complete list of network parameters, see *Network Communications Parameters*.

You can create and manage ODBC data sources on UNIX using the **iqdsn** command-line utility. See *Creating an ODBC Data Source from the Command Line*.

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. 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

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, 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-15_3/lib/libdbodbc11.so.1`. If you are creating a custom ODBC application, you can link directly to `libdbodbc11.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 see *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL Anywhere ODBC API > Building ODBC Applications > Using the SQL Anywhere ODBC driver manager on UNIX*.

Use the `libdbodbc11.so` driver and leave it up to the driver to choose the multithreaded or unthreaded driver. Tracing capability exists in the switch (`libdbodbc11.so`), not in the individual drivers (`libdbodbc11_n.so` or `libdodbc11_r.so`). If you change the driver to point to the `_r` version, you remove the switch from the call sequence, preventing the tracing.

See also

- *Ways to Avoid Shared Memory Conflicts* on page 34
- *Connection Parameters* on page 83
- *Checking Common Network Communications Problems* on page 512
- *Network Communications Parameters* on page 109
- *Creating an ODBC Data Source from the Command Line* on page 65

The iAnywhere Solutions Oracle Driver

The iAnywhere Solutions 11 - Oracle ODBC driver is custom-tailored for use with Sybase IQ or iAnywhere. This driver 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 3. iAnywhere Oracle 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.
SID	SID	The TNS Service Name that is stored in `network/admin/tnsnames.ora` under your Oracle installation directory.

Windows ODBC Administrator	Configuration for iqdsn command or .odbc.ini file	Description
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.
Encrypt Password	For **iqdsn**, use the **-pe** option.Not supported for `.odbc.ini`.	Select this option to store the password in encrypted form in the data source.
Procedures Return Results	In **iqdsn**, set the ProcResults option in the connection string.	Select this option if your stored procedures can return results. The default is that procedures do not return results (not selected). If your download_cursor or download_delete_cursor scripts are stored procedure invocations, set this to yes.
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.

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_ias_odbc_driver_for_oracle]Driver=full-
path/libdboraodbc10_r.soUserID=user-idPassword=passwordSID=TNS-
service-nameProcResults=[yes|no]ArraySize=bytes
```

2. Click **Add**.
3. Choose **iAnywhere Solutions 11 - 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 15.3 > ODBC Administrator 32-bit.
2. Click Add.
3. Choose iAnywhere Solutions 11 - 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 a Sybase IQ database using OLE DB from the following environments:

- Sybase PowerBuilder® can access OLE DB data sources, and you can use 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 Sybase IQ 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.

For more information about OLE DB, see *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL Anywhere OLE DB and ADO development* .

Note: Sybase IQ support for certain features used with OLE DB differs from the support of SQL Anywhere:

- Sybase IQ does not support Windows CE.
- Sybase IQ does not support remote updates through a cursor.
- Sybase IQ supports Dynamic (dynamic scroll), Static (insensitive), and Forward only (no-scroll) cursors, but does not support Keyset (scroll) cursors.
- In Sybase IQ the isolation level is always 3, no matter what you specify.

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 Sybase IQ:

- Sybase SQL Anywhere OLE DB provider The SQL Anywhere OLE DB provider provides access to 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.

  For more information about OLE DB providers, see *SQL Anywhere Server – Programming*.
- 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 SQL Anywhere 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 Sybase IQ 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.

For more information about ADO, including sample code for connecting with a database, see *SQL Anywhere Server – Programming*.

Note: To run examples from *SQL Anywhere Server – Programming*, substitute the data source "Sybase IQ Demo" and the samples directory `%ALLUSERSPROFILE%\All Users\SybaseIQ\samples\SQLAnywhere\VBSampler\vbsampler.sln`. Installing Sybase IQ registers appropriate drivers automatically, but you must register both the 32-bit and the 64-bit driver to develop and deploy applications based on the OLE DB on a 64-bit Windows machine. Specify the full path to the `dboledb11.dll` and `dboledba11.dll` when running regsvr32.

See also

- *Connection Parameters* on page 83

Connections From Other Databases

You can access data in Sybase IQ tables as a foreign data source from Adaptive Server Enterprise. To take advantage of this feature, you use the Component Integration Services (CIS) interface, which makes data from distributed, heterogeneous sources available to clients.

With CIS in place, you define "proxy tables" in Adaptive Server Enterprise that represent your Sybase IQ tables. You can then query the proxy tables from Adaptive Server Enterprise. For details about CIS, see *Component Integration Services User's Guide for Adaptive Server Enterprise and OmniConnect*.

CIS and Sybase IQ offer several other ways for you 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 Sybase IQ. This functionality is only supported on certain platforms. For more information, see *Installation and Configuration Guide* for your platform.
- Access data in 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 in separate Sybase IQ databases.

See *System Administration Guide: Volume 2 > Accessing Remote Data*.

See also

- *Matching Adaptive Server Enterprise Data Types* on page 291

Avoiding Port Number Conflicts on UNIX

Update configuration files to avoid port number conflicts.

1. Add the following line to `$IQDIR15/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 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

- *Security Concerns: Unrestricted Database Access* on page 78
- *Integrated Logins* on page 76
- *LogFile Connection Parameter [LOG]* on page 104

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

- *How to Test Connections* on page 75
- *Security Concerns: Unrestricted Database Access* on page 78
- *LogFile Connection Parameter [LOG]* on page 104

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** or **Integrated** (the option is case insensitive), in place of the default value of **Standard**. This step requires DBA authority).
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).

- This is the default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
- With this setting, both integrated logins and standard logins are allowed.
- With this setting, 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 DBA authority (full administrative rights).

Creating an Integrated Login in Sybase Central

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.

1. Connect to a database as a user with DBA authority.
2. Right-click the Login Mappings folder for the database, and select Login Mapping. The Integrated Login wizard is displayed.
3. On the first page of the wizard, enter the name of the system (computer) user for whom the integrated login is to be created. You can either select a name from the list or enter a name.

Also, select the database user ID this user maps to. The wizard displays the available database users. You must select one of these. You cannot add a new database user ID.

4. Follow the remaining instructions in the Wizard.

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*.

Revoking an Integrated Login Permission (SQL)

You can remove an integrated login mapping using Interactive SQL.

1. Connect to a database with DBA authority.
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 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**, 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 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 Sybase IQ database. This can be done using the Windows User Manager utility.

See also

- *How to Test Connections* on page 75
- *Integrated Logins* on page 76
- *LogFile Connection Parameter [LOG]* on page 104

Temporary Public Options Provide Added Security

Setting the value of the LOGIN_MODE option for a given database to **Mixed** 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 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*.

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 authorities granted to the newly-connected user are determined by the authorities granted to the **Guest** user ID.

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.

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.

See also

- *Logical Server Context of a Connection* on page 81

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
- In Sybase Central, a user selects Connections > Disconnect
- An application with active connections, such as Interactive SQL or Sybase Central, is closed

Users with DBA authority can also drop a specific connection:

- In Interactive SQL or Embedded SQL, by issuing a DROP CONNECTION statement
- In Sybase Central, by accessing the Users & Groups folder for the database

Connection Logging

By default, each time a user connects or disconnects from a database, the `.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*.

Connection and Communication Parameters

Connection parameters establish and describe connections from client applications to a database.

See also

- *Learning Roadmap for Connections* on page 46
- *Simple Connection Examples* on page 53
- *Sybase Central or Interactive SQL Connections* on page 49
- *Connection Strings and Character Sets* on page 398
- *Simple Connection Examples* on page 53

Connection Parameters

Connection parameters are included in connection strings.

You can enter connection parameters in the following places:

- In an application's connection string
- In an ODBC data source
- In 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 checkboxes 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 circumstances under which the parameter is to be used. Common usage entries include the following:

- Embedded databases – When 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 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 either YES, ON, 1, or TRUE if true, or NO, OFF, 0, or FALSE if false.

The connection parameters used by the interface library can be obtained from the following 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.

The following 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.

See also

- *Connections from ADO* on page 73

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 or jConnect applications such as the Java version of Interactive SQL (**dbisql**) or Sybase Central.

It 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 the following statement:

```
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 – If you specified AppInfo in the connection string, the string entered
- EXE – The name of the client executable (Windows only)
- HOST – The host name of the client machine
- IP – The IP address of the client machine
- OS – The operating system name and version number (for example, Sun Solaris 2.9)
- PID – The process ID of the client
- THREAD – The thread ID of the client
- VERSION – The version of the connection protocol in use, including major and minor values, and a build number (for example 9.0.1.1549)
- TIMEZONEADJUSTMENT – The 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 -c "uid=DBA;pwd=SQL;dbf=C:\Program Files\Sybase
  \IQ-15_3\demo\iqdemo.db"
  ```

  View the application information:

  ```
  SELECT connection_property('AppInfo')
  ```

  The result is as follows (in a single string):

  ```
  IP=10.25.99.116;
  HOST=machine-name;
  OSUSER=smith;
  OS="Windows XP Build 2600 Service Pack 3";
  EXE=C:\\Program Files\\Sybase\\IQ-15_3\\bin32\\dbisql.exe;
  PID=0x108c;
  THREAD=0x124c;
  VERSION=11.0.1.5538;
  API=iAnywhereJDBC;
  TIMEZONEADJUSTMENT=-300
  ```

- Connect to the default database from Interactive SQL, appending your own information to the AppInfo property:

  ```
  dbisql -c "uid=DBA;pwd=SQL;app=ISQL connection"
  ```

  View the application information:

  ```
  SELECT connection_property('appinfo')
  ```

The result is as follows (in a single string):

```
HOST=machine-name;
OS=Sun_Solaris 2.8;
PID=0x10e;
THREAD=0xe1;
VERSION=9.0.1.1549;
TIMEZONEADJUSTMENT=-300
APPINFO=ISQL connection
```

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 (Y) to turn on commit before execution or NO (N) to turn it off. Set this option in the .odbc.ini file or on the Advanced tab of the Connect dialog.

For example, the following causes each statement to COMMIT before execution:

```
[Sample DSN]
DatabaseFile=c:\Program Files\Sybase\IQ-15_3\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

- The following data source fragment prevents a database server from being started if no network server is located:

```
[My Demo Database]
DatabaseFile=c:\sybase\IQ-15_3\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, then 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 will not be 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-15_3\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, Sybase recommends against 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.

See also

- *Language Connection Parameter [LANG]* on page 102
- *How Connection Parameters Work* on page 47
- *Locale Information* on page 400
- *Setting a Locale* on page 400

CompressionThreshold Connection Parameter [COMPTH]

Controls the maximum size of the log file where the database server writes information about web requests.

The `CompressionThreshold` parameter is a SQL Anywhere protocol option. See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > Connection parameters and network protocol options > Network protocol options > CompressionThreshold [COMPTH].*

CommBufferSize Connection Parameter [CBSize]

Sets the maximum size of communication packets, in bytes. Use k to specify units of kilobytes.

Usage
Anywhere

Values
Integer [k]

Default
If no **CommBufferSize** value is set, the **CommBufferSize** is controlled by the setting on the server, which defaults to 1460 bytes.

Description
The **CommBufferSize** parameter specifies the size of communications packets, in bytes. The minimum value of **CommBufferSize** is 300, and the maximum is 16000.

The protocol stack sets the maximum size of a packet on a network. If you set **CommBufferSize** to be larger than that permitted by your network, the largest buffers are broken up by the network software. You should set the buffer size to be somewhat smaller than that allowed by your network, because the network software may add information to each buffer before sending it over the network. The default of 1460 allows an ethernet packet to be completely filled when using TCP/IP.

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 400 bytes:

```
...
CommBufferSize=400
...
```

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
Specify amount of space to allocate on startup for network buffers, in kilobytes.

The value 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 don't know which protocol to use.

If you specify one or more protocols in the CommLinks (LINKS) connection parameter, the client uses the named communication protocol(s), 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 as follows:

- 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.

You may wish to use a specific protocol, as opposed to **ALL**, for the following reasons:

- The network library starts slightly faster if unnecessary network links are not started.
- Connecting to the database may be faster.
- You must specify the link explicitly if you wish 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** command-line switch. By default, the network server starts all available protocols, which is equivalent to **-x ALL**. See the **-x** *list* `start_iq` utility startup option in the *Utility Guide*.

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
  ```

See also

- *Network Communications Parameters* on page 109

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
```

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.

Sybase IQ now requires the DBF parameter and the database file name to connect under special circumstances.

Usage
Embedded databases

Values
String

Default
There is no default setting.

Description
The DatabaseFile (DBF) connection parameter is used to load and connect 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 can lead to file corruption.

Example

To load and connect to the demo database (installed in directory `C:\Program Files\Sybase\IQ-15_3\demo` on Windows), use the following DBF parameter:

```
DBF=C:\Program Files\Sybase\IQ-15_3\demo\iqdemo.db
```

See also

- *DatabaseName Communication Parameter [DBN]* on page 114
- *DatabaseName Connection Parameter [DBN]* on page 93

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

There is no default setting.

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, you should specify the database name rather than the database file.

A connection will only occur 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 the following command:
  ```
  start_iq cities.db -n Kitchener
  ```
- Assuming you have run the above command, you can successfully connect to the running database named Kitchener as follows:
  ```
  DBN=Kitchener
  ```
- Alternatively, you could use the following to successfully connect to the running database named Kitchener:
  ```
  DBN=Kitchener;DBF=cities.db
  ```
- However, specifying the following would fail to connect to the database named Kitchener:
  ```
  DBF=cities.db
  ```

See also

- *DatabaseFile Connection Parameter [DBF]* on page 92
- *DatabaseName Communication Parameter [DBN]* on page 114

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

You should 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-15_3/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-15_3/demo/iqdemo.db;
dbn=my_db;dbs=-n my_db" /tmp/temp.col
```

See also

- *StartLine Connection Parameter [START]* on page 108

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
There is no default data source name.

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 Sybase IQ, Embedded SQL applications can also use ODBC data sources to store connection parameters.

Example
The following parameter uses a data source name:

```
DSN=Dynamo Demo
```

See also

- *FileDataSourceName Connection Parameter [FileDSN]* on page 99
- *ODBC Tab* on page 66
- *FileDataSourceName Connection Parameter [FileDSN]* on page 99

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 says to use 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

- The following 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

You need to supply an **EngineName** to connect to a network server. 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
```

See also

- *ServerName connection parameter [ENG]* on page 107

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 both **Password** and **EncryptedPassword** are specified, Password takes precedence.

See also

- *Password Connection Parameter [PWD]* on page 105

Encryption Connection Parameter [ENC]

Encrypts packets sent between the client application and the server.

Usage

For **ECC_TLS** (Certicom), **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 does affect performance marginally. The **Encryption (ENC)** connection parameter accepts the following arguments:

- accepts communication packets that are not encrypted. This value is equivalent to NO in previous versions of Sybase IQ.
- accepts communication packets that are encrypted with simple encryption supported on all platforms and on pre-12.6 versions of Sybase IQ. This value is equivalent to YES in previous versions of Sybase IQ.
- (formerly Certicom) accepts communication packets that are encrypted using Certicom 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 **Certicom** parameter. To authenticate the server, the Certicom software verifies that the server's certificate values match any values you supply about the client using the following arguments:
  - specify the certificate file the client uses to authenticate the server.
  - specify the value for the organization field. The server's value and the client's value must match.
  - specify the value for the organization unit field. The server's value and the client's value must match.
  - specify the certificate's common name. The server's value and the client's value must match.
- 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 Certicom software verifies that the server's certificate values match any values you supply about the client using the following arguments:

- specify the certificate file the client uses to authenticate the server.
- specify the value for the organization field. The server's value and the client's value must match.
- specify the value for the organization unit field. The server's value and the client's value must match.
- specify the certificate's common name. The server's value and the client's value must match.

Warning! The sample certificate should be used for testing purposes only. The sample certificate provides no security in deployed situations because it and the corresponding password are widely distributed with Sybase software. To protect your system, you must create your own certificate.

You can use the **connection_property** system function to retrieve the encryption settings for the current connection. The function returns one of three values: none, simple, or Certicom, depending which type of encryption is being used.

See Also

-ec *encryption-options* switch in *Utility Guide > start_iq Database Server Startup Utility*

SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Security > Transport Layer Security

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=ECC_TLS
(trusted_certificates=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=RSA_TLS
(trusted_certificates=sample.crt)"
```

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
There is no default name.

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
The following is a data source description held in a File data source:

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

See also

- *DataSourceName Connection Parameter [DSN]* on page 95
- *ODBC Tab* on page 66
- *DataSourceName Connection Parameter [DSN]* on page 95

Idle Connection Parameter [IDLE]

The Idle Connection Parameter determines when a connection will time 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 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 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 server command-line option in *Utility Guide > start_iq Database Server Startup Utility*.

Example

- The following 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

See also

For more information, see *Reference: Statements and Options > Database Options > Alphabetical List of Options > LOGIN_MODE Option*.

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, 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 or Integrated.

Examples
The following 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. The default language is the language specified by, in order, the IQLANG environment variable or the installer.

This connection parameter only affects the connection. Messages returned from SQL Anywhere's various tools and utilities appear in the default language, while the messages returned from the server appear in the connection's language.

See also

- *CharSet connection parameter [CS]* on page 88
- *How Connection Parameters Work* on page 47
- *Locale Information* on page 400
- *Setting a Locale* on page 400

LazyClose Connection Parameter [LCLOSE]

Causes the `CLOSE` *cursor-name* database request to be queued, and then 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

Note that 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 will fail. Using LazyClose is not recommended 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 sent periodically 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, the 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 the 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 handle a large number of connections more efficiently.

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

The following 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

If you want 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

The following 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
...
```

See also

- *How to Test Connections* on page 75
- *Security Concerns: Unrestricted Database Access* on page 78
- *Integrated Logins* on page 76

NewPassword Connection Parameter [NEWPWD]

The NewPassword Connection Parameter allows the user to change passwords.

Allows users to change passwords without DBA intervention, even if passwords have expired. See *SQL Anywhere 11.0.1 > SQL Anywhere Server - Database Administration > Starting and Connecting to Your Database > Connection parameters and network protocol options > Connection parameters > NewPassword connection parameter [NEWPWD]*.

Password Connection Parameter [PWD]

Provides a password for the connection.

Usage
Anywhere

Default
No password provided

Description
Every user of a database has a password. The password must be supplied for the user to be allowed to connect to the database. By default, the password has the same case sensitivity as the data. IQ databases are case sensitive by default.

The password parameter is not encrypted. If you are storing passwords in a connection profile, you should use the **EncryptedPassword** parameter. Sybase Central and the Sybase IQ ODBC configuration tool both use encrypted parameters.

If 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.

See also

- *EncryptedPassword Connection Parameter [ENP]* on page 97

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
```

See also

- *PrefetchRows Connection Parameter [PROWS]* on page 106

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;...
```

See also

- *PrefetchBuffer Connection Parameter [PBUF]* on page 105

Prompt Connection Parameter [PROMPT]

The Prompt Connection Parameter prompts the user for a new password.

Prompts the user for a new password when a user with an expired password attempts to log in and reconnects with the new password supplied.

For more information, see *SQL Anywhere Server – Database Administration*.

RetryConnectionTimeout Connection Parameter [RetryConnTO]

Instructs the client library (dblib, ODBC, ADO, and so on) to keep retrying the connection, as long as the server is not found, for the specified period of time.

See *SQL Anywhere Server – Database Administration*.

ServerName connection parameter [ENG]

Synonym for EngineName connection parameter [ENG].

See also

- *EngineName Connection Parameter [ENG]* on page 96

StartLine Connection Parameter [START]

Starts a database server running from an application.

Usage

Embedded databases

Default

No StartLine parameter

Description

You should 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.

Examples

- The following data source fragment starts a database server with a cache of 32MB.

```
StartLine=dbeng6 -c 32M iqdemo.db
```

See also

- *DatabaseSwitches Connection Parameter [DBS]* on page 94

Unconditional Connection Parameter [UNC]

Stops a server using `dbstop` even when there are connections to the server.

Usage

Anywhere

Default

No

See Also

Utility Guide > stop_iq Database Shutdown Utility

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**, then the server is shut down only if there are no active connections.

Examples

- The following command line shuts down the server unconditionally:

```
dbstop -c "uid=DBA;pwd=SQL;eng=server-name;unc=yes"
```

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.

You 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, the communications parameters are entered as the CommLinks communication parameter:

```
CommLinks=tcpip(PARM1=value1;PARM2=value2;. . .),...
```

If there are spaces in a parameter, the network communication parameters must be enclosed in quotation marks to be parsed properly by the system command interpreter:

```
start_iq -x "tcpip(PARM1=value 1;PARM2=value 2;...),..."
start_iq -x "tcpip(PARM1=value1;PARM2=value2;...)"
```

The quotation marks are required under UNIX 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.

The examples provided should all be entered on a single line; you can also include them in a configuration file and 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 & HTTPS
Broadcast [BCAST]	Certificate
BroadcastListener [BLISTENER]	Certificate_Password
ClientPort [CPORT]	DatabaseName [DBN]
DLL	LocalOnly [LOCAL]
DoBroadcast [DOBROAD]	LogFile [LOG]
Host [IP]	LogMaxSize [LSize]
LocalOnly [LOCAL]	LogOptions [LOpt]
LDAP [LDAP]	LogFormat [LF]
MyIP [ME]	MaxConnections [MaxConn]
ReceiveBufferSize [RCVBUFSZ]	MaxRequestSize [MaxSize]
SendBufferSize [SNDBUFSZ]	MyIP [ME]
ServerPort [PORT]	ServerPort [PORT]
TDS	Timeout [TO]
Timeout [TO]	
VerifyServerName [VERIFY]	

See also

- *Communications Protocols* on page 32
- *CommLinks Connection Parameter [Links]* on page 90

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).

See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > Client/server communications > Using the TCP/IP protocol*. Sybase IQ does not support the Mac OS X platform.

Example

To tell the client to broadcast only on interface number 2 when using IPv6, use:

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

See also

- *LocalOnly Communication Parameter [LOCAL]* on page 118

BroadcastListener Communication Parameter [BLISTENER]

Controls broadcast listening for this port.

Usage

TCP/IP, Server side

Values

YES, **NO**

Default
YES

Description
The **NO** value turns listening off for this port. Using **-sb 0** is the same as specifying `BroadcastListener=NO` on TCP/IP.

Example

- Start a server that accepts TCP/IP connections that use BroadcastListener=NO:

```
start_iq -x tcpip(BroadcastListener=NO)
```

Certificate Communication Parameter

Allows you to specify the name of an encryption certificate. The password for this certificate must be specified with the Certificate_Password parameter.

HTTP, HTTPS

Values
String

Default
There is no default certificate name.

Example

- Start a server that requires web connections to use a particular encryption certificate.

```
start_iq -xs
http(Certificate=cert.file;Certificate_Password=secret) ...
```

See also

- *ClientPort Communication Parameter [CPort]* on page 113
- *DoBroadcast Communication Parameter [DBROAD]* on page 115
- *Host Communication Parameter [IP]* on page 116
- *ServerPort Communication Parameter [PORT]* on page 125

Certificate_Password Communication Parameter

Allows you to specify the password that matches the encryption certificate specified by the Certificate parameter.

Usage
HTTP, HTTPS

Values
String

Default
There is no default certificate password.

Example

- Start a server that requires web connections to use a particular encryption certificate.

```
start_iq -xs
http(Certificate=cert.file;Certificate_Password=secret) ...
```

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, it is recommended that you 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.

It is best to specify a list or a range of port numbers if you wish to make multiple connections using a given Data Source or given connect string. If you specify a single port number, then your application will be able to 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 and/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 make 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
```

See also

- *Certificate Communication Parameter* on page 112
- *DoBroadcast Communication Parameter [DBROAD]* on page 115
- *Host Communication Parameter [IP]* on page 116
- *ServerPort Communication Parameter [PORT]* on page 125

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 is used to process 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
  ```

See also

- *DatabaseFile Connection Parameter [DBF]* on page 92
- *DatabaseName Connection Parameter [DBN]* on page 93

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.

For 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 it is not generally recommended.

Example

- 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
```

See also

- *Certificate Communication Parameter* on page 112
- *ClientPort Communication Parameter [CPort]* on page 113
- *Host Communication Parameter [IP]* on page 116
- *ServerPort Communication Parameter [PORT]* on page 125

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

Where

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).

See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > Client/server communications > Using the TCP/IP protocol*. Sybase IQ does not support the Mac OS X platform.

Values

String

Default

No additional machines.

Example

- 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)"
  ```

See also

- *Certificate Communication Parameter* on page 112
- *ClientPort Communication Parameter [CPort]* on page 113
- *DoBroadcast Communication Parameter [DBROAD]* on page 115
- *ServerPort Communication Parameter [PORT]* on page 125

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.

See Also

SQL Anywhere 11.0.1 > SQL Anywhere Server - Database Administration > Starting and Connecting to Your Database > Client/server communications > Using the TCP/IP protocol > Connecting using an LDAP server.

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, a server will not be autostarted.

The LocalOnly (LOCAL) communication parameter is only useful 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 will not find this server, and the Locate [**dblocate**] utility will 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.

See also

- *Broadcast Communication Parameter [BCAST]* on page 111

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

See also

- *LogMaxSize Communication Parameter [LSIZE]* on page 120
- *LogOptions Communication Parameter [LOPT]* on page 120

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 could not 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 Communication Parameter [LSIZE]

Controls the maximum size of the log file where the database server writes information about web requests.

The `LogMaxSize` parameter is a SQL Anywhere protocol option. See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > Connection parameters and network protocol options > Network protocol options > LogMaxSize protocol option [LSIZE]*.

See also

- *LogFile Communication Parameter [LOG]* on page 119
- *LogOptions Communication Parameter [LOPT]* on page 120

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, the following keywords may be used 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).

See also

- *LogFile Communication Parameter [LOG]* on page 119
- *LogMaxSize Communication Parameter [LSIZE]* on page 120

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.

See also

- *MaxRequestSize Communication Parameter [MAXSIZE]* on page 122

MaxRequestSize Communication Parameter [MAXSIZE]

Specifies the size of the largest request the database server can accept.

Description

The `MaxRequestSize` parameter is a SQL Anywhere protocol option. See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > Connection parameters and network protocol options > Network protocol options > MaxRequestSize protocol option [MAXSIZE]*.

See also

- *MaxConnections Communication Parameter [MAXCONN]* on page 121

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, 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 the keyword NONE is supplied as the IP number, no attempt is made to determine the addressing information. The NONE keyword is intended primarily for clients on operating systems where this operation is expensive, such as machines with multiple network cards or remote access (RAS) software and a network card. It is not intended for use on the server.

On Windows platforms, this option can be used 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, the interface identifier should be used. 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).

See *SQL Anywhere 11.0.1 > SQL Anywhere Server - Database Administration > Starting and Connecting to Your Database > Client/server communications > Using the TCP/IP protocol*. Note that Sybase IQ does not support the Mac OS X platform.

Example

- The following 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-15_3\demo\iqdemo.db
```

- The following 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-15_3\demo\iqdemo.db"
```

- The following 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 in order for the prefetch to occur on the open. Rebinding columns between the cursor open and the first fetch when using PreFetchOnOpen will cause reduced 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**]

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 units of kilobytes, megabytes, or gigabytes, respectively. You may want to increase the value if LOB performance over the network is important.

See also

See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Starting and Connecting to Your Database > Client/server communications > Using the TCP/IP protocol*.

The RSA security module that provides RSA encryption features to the server is provided free with Sybase IQ, but Kerberos, ECC, and FIPS security modules require a separate license. See the Installation and Configuration Guide for license details.

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.

The database server always listens on port 2638, even if you specify a different port using a network communication parameter. 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-15_3\demo\iqdemo.db
   ```

   Port number 2638 is now taken.
2. Attempt to start another database server:

   ```
   start_iq -x tcpip c:\sybase\IQ-15_3\demo\iqdemo2.db
   ```

   This fails with an error `Server failed to start. Possible cause: - port number invalid or already in use`, as the port is currently allocated.

If the **-z** switch is used as well, the console also displays the error `Couldn't bind to specified address`.

3. Start another database server, assigning a different port number to it:

```
start_iq -x "tcpip(ServerPort=2639)" c:\sybase\IQ-15_3\demo
\iqdemo2.db
```

   This should succeed 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 this low a port number, you may want to start a server that listens on an alternate port, such as 8080:

```
start_iq -xs http{port=8080} -n server3 web.db
```

See also

- *Certificate Communication Parameter* on page 112
- *ClientPort Communication Parameter [CPort]* on page 113
- *DoBroadcast Communication Parameter [DBROAD]* on page 115
- *Host Communication Parameter [IP]* on page 116

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. Sybase IQ uses these NetBIOS commands, and disallows further connections if the system has no more commands available, even if this is less than the value of the Sessions parameter.

Default

Operating system 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. If you want 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 using the TCP/IP protocol, but disallowing 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, but should be used with extreme caution. Without this limit, a rogue client could 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 when you need 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 an 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, you create a new ODBC DSN in the ODBC Administrator, and specify parameters as follows:

- On the Database tab, specify a generic server name that will be used 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, check the TCP/IP check box and type in the text box:

  ```
  host=ip_address:port#;DOBROADCAST=NONE;VERIFY=NO
  ```

  For example:

  ```
  host=123.456.77.888:2222;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.

Database Object Management

Sybase IQ lets you create, alter and delete databases and database objects such as tables, views, indexes, procedures, and login policies.

Note: Remember that Sybase IQ consists of both a catalog store and an IQ store. This chapter explains how you create both stores, and the objects in your IQ store. Tables created in the catalog store have the characteristics of SQL Anywhere tables. If you want to create tables in the catalog store, you need to refer to the SQL Anywhere documentation.

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. 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 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 Sybase IQ*.

Tools for Working with Database Objects

Sybase IQ includes three utilities for working with database objects: Sybase Central, Sybase Control Center, and Interactive SQL. In addition, PowerDesigner can be used for designing and creating whole data warehouses.

Sybase Central

Sybase Central is the primary tool for working with database objects on windowing systems.

You can use Sybase Central to manage servers, databases, and dbspaces. It lets you create, modify, and delete all kinds of database objects, including tables, procedures, views, indexes, users, and groups. To use Sybase Central on multiplex servers, see *Using Sybase IQ Multiplex*.

This chapter is concerned with the SQL statements for working with database objects. If you use Sybase Central, these SQL statements are generated for you. The primary source of information about Sybase Central is the Sybase Central online help. This chapter gives only brief pointers for tasks that you can carry out using Sybase Central.

For an introduction to using Sybase Central, see *Introduction to Sybase IQ > Managing Databases*.

Interactive SQL

Interactive SQL (**dbisql**) is a utility for entering SQL statements.

If you use Interactive SQL to work with your database schema, instead of executing the SQL statements one at a time, build up the set of commands in a **dbisql** command file. Then you can execute this file in **dbisql** to build the database.

The definitions of the database objects form the database schema. You can think of the schema as an empty database. The SQL statements for creating and modifying schemas are called the *data definition language* (DDL).

Note: Only one user at a time can perform DDL statements on a table. IQ locks a table during DDL operations on it. Users may, however, perform DDL on other objects in the same database at the same time.

If you use a tool other than Interactive SQL, all the information in these topics concerning SQL statements still applies.

Interactive SQL Command File

An Interactive SQL command file is a text file with semicolons placed at the end of commands as shown below.

```
CREATE TABLE t1 ( .. );
CREATE TABLE t2 ( .. );
CREATE LF INDEX i2 ON t2 ( .. );
..
```

An Interactive SQL command file usually carries the extension `.sql`. To execute a command file, either paste the contents of the file into the Interactive SQL command window (if the file has less than 500 lines) or enter a command that reads the file into the command window. For example, the **READ** statement:

```
read makedb
```

reads the Interactive SQL commands in the file `makedb.sql`.

See also

- *Cannot Write to a Locked Table* on page 506
- *How Locking Works* on page 376
- *Managing Write Lock Contention on a Table* on page 506

Setting Up a Sybase IQ Database

Creating an 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 Sybase IQ.

This step installs the client and server environment and the `iqdemo` database. See the *Installation and Configuration Guide* for your platform for details.

2. Create an IQ database.

This step creates both the IQ store and the catalog store. You may use Sybase Central, the SQL statement **CREATE DATABASE**, or the **iqinit** utility.

3. Create the tables in your IQ database.

Use the **CREATE TABLE** statement or the Sybase Central Create Table wizard.

4. Create indexes for the tables.

Use the **CREATE INDEX** statement or the Sybase Central Create Index wizard. You can also create certain indexes automatically when you create your tables.

5. 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.

Guidelines for Scheduling Data Definition Tasks

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

You may schedule data definition operations for times when database usage is low. All other users are blocked from reading or writing to a table while you are creating or altering that table, although for a brief time only. If the table is part of a join index, users cannot read or write to any of the tables in the join index until the data definition operation is complete. For more information on concurrency rules during data definition, see *“Locks for DDL operations”*.

See also

- *Locks for DDL Operations* on page 378
- *How Locking Works* on page 376
- *IQ PAGE SIZE Parameter Guidelines* on page 154
- *Number of Distinct Values* on page 297
- *Rules and Checks for Valid Data* on page 341

Dummy Tables for Performance Monitoring

To start the IQ buffer cache monitor you must specify a permanent or temporary table, preferably a dummy table that you use only for monitoring.

Authorities for Data Definition

In order to perform data definition tasks, you must have the appropriate authority.

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

Users with DBA or PERMS ADMIN authority and object owners can grant authority to individual users and to groups of users. You can also use the **- gu** command-line option to set the permission level required to create or delete a database.

Device Selection

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

In a production environment, raw partition installations may provide increased processing performance and better recovery capabilities. File systems, on the other hand, make it easier to manage your devices, and may be preferable in a development environment.

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

See also

- *Database File Placement* on page 149

Space Allocation

All 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 may be administered as a logical subset of total storage. You may 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. Sybase IQ has four 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.

Types of Dbspaces

Each type of dbspace stores a particular type of Sybase IQ data.

Dbspace type	Data stored	Files contained by dbspace	Number of dbspaces
The 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
IQ_SYSTEM_MAIN	IQ database structures including IQ rollforward/rollback data for each committed transaction and each active checkpointed transaction, the incremental backup metadata, and database space and identity metadata. IQ user objects may be stored here but Sybase recommends that you put them in other main dbspaces.	One or more	One

Dbspace type	Data stored	Files contained by dbspace	Number of dbspaces
Other main dbspaces (also called user dbspaces)	IQ objects such as tables, indexes, join indexes, and table metadata.	One or more	One or more
IQ_SYSTEM_TEMP	Set of 1 to n temporary dbfiles that define a single temporary dbspace for a standalone database or multiplex node	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.	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 or join index is implicitly or explicitly specified. For base tables and join indexes, 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* or **CREATE JOIN INDEX IN** *dbspace_name* clause. Base tables are typically created in a dbspace in the IQ main store, but may also be created 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.

See also

- *Data Storage in Sybase IQ* on page 3

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 as simply as the catalog.

- The SYSTEM dbspace – The IQ catalog dbspace named 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.

See also

- *Database Startup Guidelines* on page 43

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 Sybase IQ 15.3. IQ_SYSTEM_MAIN is always online when the database is open.

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488

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 to Sybase IQ 15.3

Each IQ dbspace may contain any number of files. The only limit is that the total number of IQ files is 16383.

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 Sybase IQ 15.3. 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.

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, 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. 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 become full. This ensures that your dbspaces are filled more evenly, improving disk I/O.

Create separate databases for debugging purposes. Sybase recommends that you 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, 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

Several changes in Sybase IQ 15.3 architecture affect 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 Sybase IQ 15.3 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 Sybase 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 4. 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**.
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 Sybase IQ to stripe the data across devices, which improves performance. • Only use file system files for IQ dbspaces in production if the file system is fault-tolerant and 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 a cluster file system, ideally on a Storage Area Network device.	Omit ms_size if specifying a raw device. 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 Sybase IQ sets aside in your IQ main store for certain small but critical data structures used during release savepoint, commit, and checkpoint operations.

Sybase recommends that you 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

You can 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. For more about migration, see the *Installation and Configuration Guide*.

This statement creates 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"
```

This statement creates an IQ_SYSTEM_MAIN on a raw device on Windows:

```
iqunload -au -ms_filename \\\\.\\PhysicalDrive1 -c
"UID=DBA;PWD=SQL;DBF=latest.db"
```

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sybase IQ Stops Processing or Stops Responding* on page 488

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 Sybase IQ stored procedures to estimate how much space and how many dbspaces your databases will require. See *Reference: Building Blocks, Tables, and Procedures > System Procedures* for syntax and usage notes for each procedure.

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. Add totals under "RAW DATA index_size" for all tables together.
3. Run the stored procedure **sp_iqestjoin** to estimate the amount of additional space required to create join indexes on tables that you want to join frequently. Run the procedure once for each pair of tables, as follows: **sp_iqestjoin** *table1*, *table1rows*, *table2*, *table2rows* [,*relation*] [,*iqpagesize*] ...

   sp_iqestjoin suggests different index sizes depending on your queries.

   Each time you run **sp_iqestjoin**, select one of the suggested index sizes. If you know you will always join the tables with exact one-to-one matches, use the "Min Case index_size". If you anticipate occasional one-to-many joins, use the "Avg Case index_size". If you anticipate using numerous one-to-many joins, use the "Max Case index_size".
4. Add together the index_sizes you selected for all table pairs.
5. Add the join space total from step number 4 to the table space total from step number 2, performing a separate calculation for minimum and maximum join space.
6. 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.

See also

- *Disconnecting Other Users From a Database* on page 160

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.)

Range Partitions

Partitioning is a scheme of dividing large objects into subobjects.

For example:

- Storage space is partitioned into tablespaces
- Tables can be partitioned into table partitions

Table objects can be table partitions, columns, indexes, IQ base tables, join indexes, unique constraints, primary keys or foreign keys.

A *table partition* is a collection of rows that is a subset of a user-created table. A given row cannot be placed in two different partitions. Each partition can be placed in its own dbspace and managed individually. A partition shares its parent table's logical attributes:

- Column definition with the same integrity constraints and defaults
- The same referential integrity constraints
- The same unique and primary key constraints
- The same check constraints

The table creator chooses whether to partition a table, how to partition it, and the number of partitions. The table creator defines the *partition key*, a table column that determines how a table should be partitioned. See *Reference: Statements and Options > SQL Statements> CREATE TABLE statement* and *Reference: Statements and Options > SQL Statements> ALTER TABLE statement* for syntax.

A fundamental administration concept of partitioning is the ability to make a subobject read-only. Once a subobject is set read-only, validated, and backed up, it needs minimal maintenance.

Benefits of Dbspace Management and Table Partitioning

Partitioning and dbspace management allow administrative operations (data placement, dbcc, backup, restore) to be performed at a finer granularity than at the table or database level.

In Sybase IQ 15.3, dbspace management and range table partitions:

- Provide data placement capability
- Provide hierarchical storage management by supporting relocation of less critical data to cheaper storage

Dbspace management and range table partitions improve maintainability and availability by:

- Supporting dbspace read-only (RO) vs. read-write (RW) status
- Supporting dbspace online vs. offline status

- Shortening backup/restore by allowing backing up or restoring one or more RO dbspaces and/or files, or all RW files
- Supporting data validation on a dbspace or a table partition target. (See *Reference: Building Blocks, Tables, and Procedures > System Procedures > sp_iqcheckdb procedure*.)
- Allowing Sybase IQ servers to start with unavailable and/or non-usable dbspaces marked “offline” except for the catalog store and the Sybase IQ system dbspace.

Partitioning improves administration and runtime operations, particularly backup, restore, and database validation, by organizing storage and data according to business requirements.

Sybase IQ 15.3 *range partitioning* divides a table into logical partitions based on the values of a single table column. Only base tables can be partitioned; global temporary tables or declared local temporary tables cannot. All rows of a table partition are physically colocated, and the user must name each partition.

Sybase IQ supports a maximum of 1024 partitions for range partitioning.

The most common form of range partitioning is to partition the table by date; for example, June_2009, July_2009, and so on. A range table partition may be assigned to a dbspace.

Restrictions on DDL Operations on Partitions

You can drop, rename, partition, unpartition, merge, split, and move table partitions, with the following restrictions:

Operation	Restriction
Drop	You cannot drop a partition key column or the last partition of a partitioned table.
Rename	N/A
Partition an unpartitioned table	N/A
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 one of the two partitions after splitting. Split partition must be on same dbspace as original so that no data movement is required.
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.

Partitioned tables cannot participate in a join index.

See also

- *Considerations for Partitioned Table Loads* on page 258

DML Operations on Partitions

You can perform DML operations including load, insert, delete, truncate, and truncate table partition for range partitioning. Update is supported except updating the partition key column.

Sybase IQ returns an error 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.

The `START ROW ID` option for load on a partitioned table is disallowed. The `APPEND_LOAD` option for loads on a partitioned table appends new rows to the end of the appropriate partition.

See also

• *Considerations for Partitioned Table Loads* on page 258

Object Placement for Non-Partitioned Tables

You can specify a dbspace for a table object (including columns of any data type, indexes, primary and foreign keys, unique constraints, join indexes, and non-partitioned tables) at object creation or when you move the object.

You must have CREATE privilege in the dbspace. For example:

```
CREATE TABLE tab1 (
        col1 INT IN dsp1,
        col2 VARCHAR(20),
        col3 CLOB IN dsp2,
        col4 DATE,
        col5 BIGINT,
        PRIMARY KEY (col5) IN dsp4) IN dsp3;
CREATE DATE INDEX col4_date ON tab1(col4) IN dsp5;
```

Resulting data allocation is as follows:

Dbspace name	Data
dsp1	FP index for col1
dsp2	FP index for col3
dsp3	FP indexes for col2, col4, and col5
dsp4	Primary key (HG on col5)
dsp5	DATE index col4_date

Object Placement for Partitioned Tables

For a partitioned table, you can place each table partition in an individual dbspace.

You may also place each column for a table partition in an individual dbspace. In general, however, individual dbspaces are recommended only for `BLOB` or `CLOB` columns or columns of `CHAR`, `VARCHAR` or `VARBINARY` greater than 255 bytes. For example:

```
CREATE TABLE tab2(
    col1 INT IQ UNIQUE(65500),
    col2 VARCHAR(20),
    col3 CLOB PARTITION (p1 IN dsp11, p2 IN dsp12,
        p3 IN dsp13),
    col4 DATE,
    col5 BIGINT,
    col6 VARCHAR(500) PARTITION (p1 IN dsp21,
        p2 IN dsp22),
    PRIMARY KEY (col5) IN dsp2) IN dsp1
    PARTITION BY RANGE (col4)
    (p1 VALUES <= ('2006/03/31') IN dsp31,
    p2 VALUES <= ('2006/06/30') IN dsp32,
    p3 VALUES <= ('2006/09/30') IN dsp33
    );CREATE DATE INDEX c4_date ON tab2(col4) IN dsp3;
```

Resulting data allocation is as follows:

Partition	Dbspace	Data
p1	dsp11	FP indexes for col3 (CLOB data)
	dsp21	FP index for col6 (VARCHAR(500) data)
	dsp31	FP indexes for col1, col2, col4, and col5
p2	dsp12	FP for col3 (CLOB data)
	dsp22	FP for col6 (VARCHAR(500) data)
	dsp32	FP indexes for col1, col2, col4, and col5
p3	dsp13	FP index for col3 (CLOB data)
	dsp33	FP indexes for col1, col2, col4, col5, and col6 (varchar(500) data)
Non-partitioned	dsp1	Lookup store for col1 and other share data (for all partitions)

Partition	Dbspace	Data
Non-partitioned	dsp2	Primary key HG on col5 (for all partitions)
Non-partitioned	dsp3	DATE index col4_date (for all partitions)

Database Definition

To define your database, you can use SQL statements or you can use 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 Central

To create a Sybase IQ database in Sybase Central, see the *Sybase Central Plug-in Help*.

Database Creation with Sybase Control Center

To create a Sybase IQ database in Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

Multiplex Database Creation

To create multiplex databases, see *Using Sybase IQ Multiplex > Multiplex Creation > Converting Databases to Multiplex*.

Setting Database Options with Sybase Control Center

For instructions on using Sybase Control Center to set database options, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=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.

See also

- *CREATE DATABASE Statement Defaults* on page 150
- *Multiplex Capability* on page 5
- *Relative Path Names* on page 151
- *Utility Database Server Security* on page 314

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 any 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:

   ```
   dbisqlc -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.)
3. In Sybase Central, in the Create Database wizard, choose Use the IQ Agent on a Remote Host to Start a Utility Server.

See also

- *Utility Database Server Security* on page 314

Dbspaces Created Automatically

When you create a database, the database server creates the following four dbspaces:

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.msg*
`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.

Warning! When you allocate file 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, Sybase recommends that dbspace management be performed by a single database administrator on a single connection.

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 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 Adaptive Server Enterprise.

If your IQ requirements are large and complex enough that you need multiple physical systems, consider using 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/`*`rdevname`*. For example, `/dev/raw/raw10`.

Database File Compatibility

Sybase IQ servers cannot manage databases created with versions prior to Sybase IQ 12.6; likewise, old servers cannot manage new databases.

See also

- *System Recovery and Database Repair* on page 453

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.

See also

- *Database Creation with SQL* on page 147
- *Multiplex Capability* on page 5
- *Relative Path Names* on page 151
- *Utility Database Server Security* on page 314

Relative Path Names

You can create a database using a relative or fully qualified path name for each of the files for the database.

Sybase recommends that you 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, as described in as described in *Reference: Statements and Options > SQL Statements> CREATE DATABASE statement*.

If you omit the directory path, 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.

See also

- *CREATE DATABASE Statement Defaults* on page 150
- *Database Creation with SQL* on page 147
- *Multiplex Capability* on page 5
- *Utility Database Server Security* on page 314

IQ PATH Parameter Guidelines

The required **IQ PATH** parameter tells Sybase IQ that you are creating an IQ database, not an 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 5. 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 6. Raw device names on Windows

Device type	Name format required	Example
Partitioned	Letter assigned to that partition	`\\.\C:` in Sybase Central, `\\\\.\\C:` in SQL
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` in Sybase Central, `\\\\.\\ PhysicalDrive32` in SQL

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/iqdata/company.iqtmp`
- The IQ message log file is in `/disk1/company/iqdata/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, Sybase recommends 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.

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.

See also

- *Page Size* on page 17
- *Maximum Catalog Page Size* on page 31
- *Guidelines for Creating Tables* on page 179
- *How Locking Works* on page 376
- *Number of Distinct Values* on page 297
- *Rules and Checks for Valid Data* on page 341

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**; 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 7. 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
262144	16384000	8192000	16MB	8MB
524288	32768000	16384000	32MB	16MB

PAGE SIZE Parameter Guidelines

You can select a page size for the catalog store 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 4KB, where the IQ store is on a UNIX raw partition and has an **IQ PAGE SIZE** of 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.

Java Support Guidelines

By default, Java support is ON for IQ databases.

It can be turned off with the **JAVA OFF** option. With Java ON:

- You can write a Java procedure that accesses tables in the catalog store or the IQ store. These queries are processed like any other query.
- You cannot store Java data in an IQ table or a catalog store table. If you attempt to create an IQ column of type Java, you receive an error.
- Java application programming Interface (API) can be used in stored procedures.
- The JDBC interface can be used with to access SQL data only, because you cannot store Java data in any table in an IQ database.
- Sybase IQ has been certified with the combined Java/Stored Procedure debugger, which is supplied on the Network Client CD.

These Java features are supported in the catalog store only:

- You cannot use a Java-based user-defined function within a query to an IQ table, but you can use it on catalog store tables.
- You cannot use Java classes as data types in IQ tables, but you can use Java classes as data types in catalog store tables.
- The Java API classes supported by SQL Anywhere are also supported in the IQ catalog store.

Sybase IQ supports access to SQL Anywhere tables from IQ, and to IQ tables from SQL Anywhere, by means of proxy tables. Additional Java features should work when you use remote data access capabilities to access IQ tables from Anywhere, or Anywhere tables from IQ:

- You can use Java-based user-defined functions in queries on tables in an SQL Anywhere database, or queries to IQ tables from an SQL Anywhere database. For details on using remote data access capabilities, see *System Administration Guide: Volume 2 > Accessing Remote Data*.
- You can include Java operations in a SQL statement.
- You can use Java API classes in SQL statements.
- You can treat the Java API classes as extensions to the available built-in functions provided by SQL.

For details on Java support in Sybase IQ, see:

- *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > Java in the database > Java support in SQL Anywhere > Introduction to Java support*.
- *System Administration Guide: Volume 2 > Debugging Logic in the Database*.

See also

- *How Sybase IQ Establishes Connections* on page 46
- *Default Connection Parameters* on page 61
- *Connecting From Sybase IQ Utilities* on page 62

Setting Database Options in Sybase Central

Database options are configurable settings that change the way the database behaves or performs.

1. Open the server.
2. Right-click the database and choose Options.
3. Edit the values.

Note: With the Database Options dialog, you can also set database options for specific users and groups.

When you set options for the database itself, you are actually setting options for the PUBLIC group in that database, because all users and groups inherit option settings from PUBLIC.

- See *Reference: Statements and Options > Database Options*.

See also

- *Resource Issues* on page 495

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 group in that database, because all users and groups inherit option settings from PUBLIC.

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 DBA authority can limit the amount of space used per connection. In addition, when 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 a 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.

See *Reference: Statements and Options > Database Options*.

See also

- *Insufficient Disk Space* on page 495
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488
- *Resource Issues* on page 495
- *Finding the Currently Executing Statement* on page 521
- *Logging Server Requests* on page 522
- *Sybase IQ Stops Processing or Stops Responding* on page 488

Showing System Objects in Sybase Central

In a database, a table, view, stored procedure, or domain is a system object.

1. Open the server.
2. Right-click the connected database and choose Configure Owner/Container Filtering.
3. Enable **SYS** and **dbo**, and click OK.

   The system tables, system views, system procedures, and system domains appear in their respective folders (for example, system tables appear alongside normal tables in the Tables folder).The system views are owned by the SYS user ID.

See also

- *Optimizing Storage and Query Performance* on page 183

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
   ```

Disconnecting from a Database in Sybase Central

When you are finished working with a database, you can disconnect from it.

1. Open the desired server.
2. Select the desired database.
3. On the toolbar, click the Disconnect button.

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
```

See also

- *Data Storage* on page 161
- *Dbspace Management Example* on page 172
- *Disk Space* on page 16
- *Locks for DDL Operations* on page 378
- *Versioning of Temporary Tables* on page 375

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
```

See also

- *Dbspace Management Permissions* on page 309

Disconnecting Other Users From a Database

Sybase IQ also 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 with DBA authority.
2. Using Interactive SQL, execute a **DROP CONNECTION** statement.

   The following statement drops the connection with ID number 4.

   ```
   DROP CONNECTION 4
   ```

See also

- *Estimating Space and Dbspaces Required* on page 141

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.

Data Storage

A DBA can determine which tables, indexes, and join 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.

See also

- *Dbspace Management Example* on page 172
- *Disk Space* on page 16
- *Locks for DDL Operations* on page 378
- *Versioning of Temporary Tables* on page 375
- *Disconnecting All Connections from a Database in Interactive SQL* on page 159

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 Reference: Statements and Options > SQL Statements> ALTER DBSPACE Statement.

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.

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.

Operations that can be performed on dbspaces include adding, dropping, and renaming. See *Reference: Statements and Options > SQL Statements > CREATE DBSPACE Statement*, *Reference: Statements and Options > SQL Statements > DROP Statement*, and *Reference: Statements and Options > SQL Statements > ALTER DBSPACE Statement*.

For multiplex dbspaces, see *Using Sybase IQ Multiplex > Multiplex Transactions*. To change the status of a dbspace in Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=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	Table's dbspace	Table's status
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 or join index 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.) Sybase IQ returns an error for DML operations on a read-only table or read-only table partition.

Operations to join indexes, including creating, dropping and synchronizing join indexes, fail if any of the join tables are read-only.

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 8. Allowed dbspace configuration transformations

State	Alter type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYS-TEM_TEMP
Online dbspace			
	ALTER DBSPACE OFFLINE	Yes, if RO	No
	ALTER DBSPACE ONLINE	No	Yes for IQ_SYS-TEM_MAIN, No for IQ_SHARED_TEMP and IQ_SYSTEM_TEMP
	ALTER DBSPACE READON-LY	Yes, if it is RW	No
	ALTER DBSPACE READ-WRITE	Yes, if it is RO	No
	ALTER STRIPING or STRIPE-SIZEKB	Yes	Yes on simplex and on multiplex coordinator
	RENAME DBSPACE	Yes	No
	ADD FILE	Yes	Yes
	DROP FILE	Yes, if empty and RO	Yes, if empty and RO. Dropping files in IQ_SHARED_TEMP re-quires that this operation be done only on the coor-dinator once it is started in a single node mode. Drop-ping 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

State	Alter type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYS-TEM_TEMP
	ALTER FILE SIZE	Yes, if RW dbspace and dbfile	Yes, if RW
	ALTER FILE RENAME LOG-ICAL 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_SYS-TEM_MAIN.
Offline dbspace			
	ALTER DBSPACE OFFLINE	No	N/A
	ALTER DBSPACE ONLINE	Yes	N/A
	ALTER DBSPACE READON-LY	No	N/A
	ALTER DBSPACE READ-WRITE	No	N/A
	ALTER STRIPING or STRIPE-SIZEKB	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 LOG-ICAL NAME	Yes	N/A
	ALTER FILE RENAME PATH	Yes	N/A
Dynamically offline dbspace			

State	Alter type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYS-TEM_TEMP
	ALTER DBSPACE OFFLINE	Yes, if RO	N/A
	ALTER DBSPACE ONLINE	Yes	N/A for IQ_SYS-TEM_MAIN and IQ_SYSTEM_TEMP, Yes for IQ_SHARED_TEMP
	ALTER DBSPACE READON-LY	Yes, if RW	N/A
	ALTER DBSPACE READ-WRITE	No	N/A
	ALTER STRIPING or STRIPE-SIZEKB	Yes	N/A
	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 LOG-ICAL 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 READON-LY	No	N/A
	ALTER DBSPACE READ-WRITE	Yes, if online	N/A
	ALTER STRIPING and STRI-PESIZEKB	Yes	N/A

State	Alter type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYS-TEM_TEMP
	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 LOG-ICAL NAME	Yes	N/A
	ALTER FILE RENAME PATH	Yes, if offline	N/A
Read-write dbspace			
	ALTER DBSPACE OFFLINE	No	No
	ALTER DBSPACE ONLINE	Yes, if dynamical-ly offline	No for IQ_SYS-TEM_MAIN and IQ_SYSTEM_TEMP, Yes if dynamically offline for IQ_SHARED_TEMP
	ALTER DBSPACE READON-LY	Yes	No
	ALTER DBSPACE READ-WRITE	No	No
	ALTER STRIPING and STRI-PESIZEKB	Yes	Yes
	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_SYS-TEM_TEMP
	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.
	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			

State	Alter type	Allowed for User Main	Allowed for IQ_SHARED_TEMP, IQ_SYSTEM_MAIN, or IQ_SYSTEM_TEMP
	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

Notes

- 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'.`

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 or the Sybase Central Create Dbspace wizard.

A new dbspace can be on the same or a different disk drive as the existing dbspaces. You must have DBA or SPACE ADMIN authority.

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.

Sybase recommends that you create main stores on raw devices.

When you specify a raw device for a new dbspace, 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, 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 Central, see the online help. To create a dbspace in Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

After you add or drop a dbspace, issue a **CHECKPOINT**. System recovery begins after the most recent checkpoint.

See also

- *Connecting to the demo database from Sybase Central* on page 53
- *Connecting To the Demo Database From Interactive SQL* on page 54
- *Creating Foreign Keys* on page 187

Guidelines for Dropping a Dbspace

Issue a **DROP DBSPACE** command to remove a database file.

To drop a dbspace:

- You must have DBA or SPACE ADMIN authority.
- It cannot contain any data from user tables or join indexes. 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 Sybase IQ fills dbspaces with data, it is unlikely that a dbspace will become empty only after explicitly relocating tables and join indexes, 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, join index, 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 Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

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	50M	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	DBSpaceName	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	DBSpaceName	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*.

See also

- *Data Storage* on page 161
- *Disk Space* on page 16
- *Locks for DDL Operations* on page 378
- *Versioning of Temporary Tables* on page 375
- *Disconnecting All Connections from a Database in Interactive SQL* on page 159

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 Sybase Central or Interactive SQL.

Table Creation with Sybase Central

To create a table using Sybase Central, see the online help.

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, Sybase IQ makes permanent all previous changes to the database. There is also a **COMMIT** after the table is created.

For a full description of the **CREATE TABLE** statement, see *Reference: Statements and Options*.

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.

See also

- *How Locking Works* on page 376
- *IQ PAGE SIZE Parameter Guidelines* on page 154
- *Number of Distinct Values* on page 297
- *Rules and Checks for Valid Data* on page 341

Types of Tables

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**, or by using the Global Temporary Table Creation wizard in Sybase Central.

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. See *Using Sybase IQ Multiplex > Multiplex Transactions > DML Commands > Table Data Scope*.

To select into a temporary table, use syntax like the following:

```
SELECT * INTO #TableTemp FROM lineitem
WHERE l_discount < 0.5
```

Note: 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, or with Sybase Central.

You cannot drop or alter a global temporary table while other connections are using the table.

See also

- *Using Join Indexes* on page 223

Placement of Tables

Sybase IQ creates tables in your current database.

If you are connected to an IQ database, tables are placed as follows:

Table 9. 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

Join Virtual Tables

A Join Virtual Table is a denormalized table that looks like a regular table; it has a name, columns, rows, and indexes.

Sybase IQ creates Join Virtual Tables as a result of a **CREATE JOIN INDEX** for internal processing purposes and deletes them when you do a **DROP JOIN INDEX**. You cannot create, modify, or delete Join Virtual Tables, but you may see error messages related to them if you try to use or modify them. Sybase suggests that you ignore all Join Virtual Tables.

Servers running in a multiplex cannot create or drop join indexes. For more information, see *Using Sybase IQ Multiplex*.

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 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.

See also

- *Overview of Indexes* on page 195
- *Criteria for Choosing Indexes* on page 200

Optimizing Storage and Query Performance

When you create a permanent table in an IQ database, Sybase IQ automatically stores it in a default index that facilitates a type of query called a projection.

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 `MINIMIZE_STORAGE` option (**SET OPTION** or plug-in Database Options dialog)
- The data type of the column and its width
- The **IQ PAGE SIZE** option (**CREATE DATABASE** or plug-in Create Database wizard)

See also

- *Showing System Objects in Sybase Central* on page 159

Effect of IQ UNIQUE

See the following table for implications of the **CREATE TABLE** option **IQ UNIQUE**.

Table 10. Effect of IQ UNIQUE

IQ UNIQUE 256 or less	IQ UNIQUE 65536 or less	IQ UNIQUE 16777216 or less	IQ UNIQUE unspecified or greater than 16777216
Storage optimized for small number of unique values	Storage optimized for medium number of unique values	Storage optimized for 3-byte FP indexes	Storage optimized for large number of unique values
Faster query performance, less main IQ store space required	Faster query performance, less main IQ store space required	Faster query performance, less main IQ store space required	Queries may be slower
Need a small amount of extra cache for IQ temporary store for loads	Need extra cache for IQ temporary store for loads. The amount depends on the number of unique values and the data type.	Need significant extra cache for IQ temporary store for loads. The amount depends on the number of unique values and the data type.	No extra cache needed for loads

IQ UNIQUE 256 or less	IQ UNIQUE 65536 or less	IQ UNIQUE 16777216 or less	IQ UNIQUE unspecified or greater than 16777216
Loads may be slower if you have numerous columns with IQ UNIQUE <256	Loads may be slower	Loads may be slower	Loads are faster

Effect of MINIMIZE_STORAGE Option

When `MINIMIZE_STORAGE` is ON, it is equivalent to specifying **IQ UNIQUE 255** for all new columns.

`MINIMIZE_STORAGE` defaults to OFF. For details, see *Reference: Statements and Options > Database Options > General Database Options > MINIMIZE_STORAGE Option*.

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. For details, see *Reference: Building Blocks, Tables, and Procedures > System Procedures*.

Difference Between UNIQUE and IQ UNIQUE

IQ UNIQUE (count) gives an approximation of the number of distinct values that can be in a given column. 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.

For a complete description, see *Reference: Statements and Options > SQL Statements > ALTER TABLE Statement*.

Altering Tables in Sybase Central

To alter a table definition in Sybase Central, see the Sybase Central online help.

Altering Tables in a Join Index

You cannot **ADD**, **DROP** or **MODIFY** a base table column that participates in a join condition of a join index. To alter joined columns, you must first drop the join index, alter the table, and then recreate the join index. See *"Using join indexes"* for complete information on join indexes.

See also

- *Table and Column Constraints* on page 351

Guidelines for Dropping Tables

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, except if any column in the table participates in a join index.

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; see Chapter 10, *"Transactions and Versioning"* for details.

DROP TABLE and **TRUNCATE TABLE** statements execute in seconds. The size of the data does not affect the speed of the operation.

For a full description see *Reference: Statements and Options > SQL Statements > DROP Statement*.

Dropping Tables in Sybase Central

You can execute **DROP TABLE** statements in Sybase Central or Interactive SQL.

1. Connect to the database.
2. Click the Tables folder for that database.
3. Right-click the table you wish to delete, and select Delete from the pop-up menu.

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.

See also

- *Connecting to the demo database from Sybase Central* on page 53
- *Connecting To the Demo Database From Interactive SQL* on page 54
- *Creating Foreign Keys* on page 187

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*.

For more information about using primary and foreign keys, see Chapter 9, "Ensuring Data Integrity"

See also

- *Additional Dbspaces* on page 169
- *Creating Primary Keys* on page 186
- *Entity and Referential Integrity* on page 355
- *Star Joins* on page 234

Table Information in System Views

Information about tables in a database is in the system view SYS.SYSTAB.

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.SYSTAB view:

```
SELECT *
FROM SYS.SYSTAB
```

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.

See also

- *Views and Procedures Provide Extra Security* on page 332

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.

To create a view in Sybase Central, see the online help.

Guidelines for Using Views

When you use views, you need to be aware of certain restrictions, both on the **SELECT** statements you can use to create them, and on your ability to insert into, delete from, or update them.

Restrictions on SELECT Statements

There are some restrictions on the **SELECT** statements that you can use as views. In particular, 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 would impose an order on the rows of the view. You can use the **GROUP BY** clause, subqueries, and joins in view definitions.

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 **SELECT** query just right, 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 in the following cases:

- Views containing aggregate functions, such as **COUNT(*)**
- Views containing a **GROUP BY** clause in the **SELECT** statement
- Views containing 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. Deleting such views or changing them into tables may cause unexpected problems.

See also

- *Views Provide Tailored Security* on page 332

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
```

See also

- *Restrictions on Modifying Join Index Tables* on page 237

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 base tables 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 base table(s) referred to by the view, or
  - a non-DBA, and not the owner of some or all of the base table(s) referred to by the view, but the view owner has SELECT permission WITH GRANT OPTION on the base table(s) not owned and any other required permission(s) WITH GRANT OPTION on the base table(s) not owned for the operation.

    Instead of the owner having permission(s) WITH GRANT OPTION on the base table(s), 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
```

How to Drop a View in Sybase Central

To drop a view in Sybase Central, right-click the view you wish to delete and select Delete from the pop-up menu.

For more information, see the Sybase Central online Help.

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
```

Index Management

Sybase IQ indexes dramatically improve the performance of database searches over searches in traditional relational databases.

Performance is a vital consideration when designing and creating your database.

Even within Sybase IQ, however, it is important to choose the right indexes for your data, to achieve the greatest performance, and to make best use of memory, disk, and CPU cycles.

Introduction to Indexes

All IQ database columns with data need an index.

When you create a database in an IQ store, a default index is created automatically on every column of every table. You can also choose from several other index types:

- Four column index types optimize specific types of queries on the indexed column.
- Join indexes optimize queries that relate columns from two or more tables.

You will almost certainly want to supplement the default indexing by selecting one or more indexes for many of the columns in your database. You will also want to define join indexes for any table columns that are joined in a consistent way in user queries. Select indexes based on the size of your database, the disk space available, and the type of queries users submit.

Indexes are created on a specified table, or on a set of tables for join indexes. You cannot create an index on a view.

Creating Indexes

Create column indexes and join indexes.

You can create column indexes in three ways:

- With the **CREATE INDEX** statement
- With the Index Creation wizard in Sybase Central
- With the **UNIQUE** or **PRIMARY KEY** column constraint of **CREATE TABLE**, which creates a unique index automatically.

You can create a join index in two ways:

- With the **CREATE JOIN INDEX** statement
- With the Create Join Index wizard in Sybase Central

Index Information in System Views

Information on indexes is in the system views `SYSINDEX`, `SYSIQINDEX`, `SYSIXCOL`, and for join indexes, `SYSIQJOINIXTABLE`.

See *Reference: Building Blocks, Tables, and Procedures > System Tables and Views* for a description of these views.

Displaying Indexes Using Stored Procedures

You can also use the stored procedure **sp_iqindex** to display a list of indexes and information about them.

For example, to list the indexes in the `Departments` table, issue the command:

```
sp_iqindex 'Departments'
```

The following information displays. Output is displayed in two pieces for readability:

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

If you omit the table name from the command, **sp_iqindex** displays this information for all tables in the database.

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 Validation

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.

For foreign key indexes, a validation check also ensures that the corresponding row exists in the primary table, and that their hash values match. This check is equivalent to the validity checking carried out by the SQL Anywhere **VALIDATE TABLE** statement.

To validate an index, open a command prompt and run the **dbvalid** utility.

For example, the following statement validates an index called EmployeeIndex. The **-i** switch specifies that each object name given is an index. (Type the command on one line.)

```
dbvalid -c "uid=dba;pwd=sql;eng=myserver"
-i EmployeeIndex
```

See *Utility Guide > dbvalid Database Administration Utility*.

How to Rename Indexes

You can rename an index in a base table or global temporary table with the owner type USER.

See *Reference: Statements and Options > SQL Statements > ALTER INDEX Statement* for more information on renaming indexes and changing foreign key role names. Note that indexes created to enforce key constraints cannot be renamed.

How to Remove Indexes

If a column index or join index is no longer required, you can remove it from the database using the **DROP** statement.

You can also drop indexes in Sybase Central by clicking the table name, right-clicking to display options, and clicking the Delete option.

See also

• *Restrictions on Modifying Join Index Tables* on page 237

Sybase IQ Indexes

Before you index your Sybase IQ data, you must understand Sybase IQindex types, how you create an index, how you decide what index types are best suited for the way you use the data in your database, and performance and resource issues related to indexing.

Overview of Indexes

Indexes improve data retrieval performance. Sybase IQ indexes differ from traditional indexes and are designed to accelerate queries in the data warehousing environment.

Traditional indexes often use a B-tree index strategy to point to the data records. That strategy is valuable only if many unique data values are used to filter down to a very small set of records, as with columns of order numbers or customer names, as you would encounter in a transaction-processing system.

Sybase IQ indexes actually represent and store the data so that the data can be used for accelerating a wide variety of queries. This strategy is designed for the data warehousing environment, in which queries typically examine enormous numbers of records, often with relatively few unique values, and in which aggregate results are commonly required.

Sybase IQ Index Types

A *default index* that optimizes projections is created by Sybase IQ for all columns.

Depending on the size of your database, the disk space available to you, and the type of queries your users submit, you almost certainly want to supplement the default index with one or more of the Sybase IQ bitwise index types.

When you load data into a table, 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.

Logically, the data can still be accessed row-by-row, just as in more traditional row-based SQL databases. Column storage structures your data according to the attributes you are interested in tracking. In a data warehousing environment, you typically look at specific attributes of thousands or millions of rows of data, rather than complete, single rows of data that are traditionally the focus in transaction processing. Column storage optimizes your ability to perform selections or calculations on the attributes you care about.

The default column storage structure that Sybase IQ creates for each column is actually an index optimized for storing and projecting data. The column indexes you define are created as part of each individual table. Create join indexes with care; they add significant load, update, and delete costs.

Besides column indexes, Sybase IQ lets you define join indexes. Join indexes are optimized for joining related tables. You may want to create a join index for any set of tables that your users commonly join to resolve queries. Column indexes underlie any join indexes involving those columns.

Columns with fewer than 16 million unique values can be stored in an optimized default index that significantly reduces storage requirements. This format supports improved performance by the IQ optimizer and for the aggregate functions **SUM**, **SUM DISTINCT**, **MAX**, **MIN**, and **COUNT DISTINCT**. It is available for:

- Any column where **IQ UNIQUE()** is specified
- All columns created when the `MINIMIZE_STORAGE` database option is ON

To achieve maximum query performance, however, you should choose one or more additional index types illustrated in the following table for most columns that best represent the cardinality and usage of column data:

Table 11. Sybase IQ index types

Index type	Description
Compare or **CMP**	Stores the binary comparison (<, >, or =) of any two distinct columns with identical data types, precision, and scale.
DATE	An index on columns of data type `DATE` used to process queries involving date quantities.
Datetime or **DTTM**	An index on columns of data type `DATETIME` or `TIMESTAMP` used to process queries involving datetime quantities.
High_Group or **HG**	An enhanced B-tree index used to process equality and group by operations on high-cardinality data (recommended for more than 1,000 distinct values).
High_Non_Group or **HNG**	A non value-based bitmap index ideal for most high-cardinality decision support operations involving ranges or aggregates.
Low_Fast or **LF**	A value-based bitmap index for processing queries on low-cardinality data (recommended for up to 1,000 distinct values but can support up to 10,000 distinct values.)
TIME	An index on columns of data type `TIME` used to process queries involving time quantities.
WD	Used to index keywords by treating the contents of a `CHAR`, `VARCHAR`, or `LONG VARCHAR` column as a delimited list.
TEXT	Used to index terms (words) and their positions. Provides ability to search for individual terms, phrases, pairs of terms within specified distances, and given order, as well as combinations of these conditions.

Select column indexes according to the type of data in the column and your intended operations for the column data. In general, you can use any index or combination of indexes on any column. However, there are some exceptions.

When a table is created with the `DATE` data type, an optimized two-byte FP index is created on the `DATE` field, which is independent of the settings in database option `MINIMIZE_STORAGE`.

If you want to create a three-byte FP or flat-style FP index on the `DATE` field, use the following `IQ UNIQUE` values when creating the table:

- For a three-byte FP — `IQ UNIQUE` should be between 65537 and 16777216.
- For flat-style FP — `IQ UNIQUE` should be higher than 16777216.

To take advantage of the High_Non_Group index types for columns with nonintegral numeric data, use the `NUMERIC` or `DECIMAL` data types, which support up to 254 digits of precision.

Some index types have duplicate functionality; creating unnecessary indexes wastes disk space. Read the sections that follow for details on how to select an index.

When a column is designated as **FOREIGN KEY**, **PRIMARY KEY**, or **UNIQUE**, Sybase IQ creates a High_Group index for it automatically. For each foreign key, Sybase IQ creates a non-unique High_Group index.

Note: You can also create a High_Group index on a set of columns explicitly. For details, see the **CREATE INDEX** statement in *Reference: Statements and Options*.

How Sybase IQ Uses Indexes

You may also want to define additional indexes on your columns for best performance. Sybase IQ uses the fastest index available for the current query or join predicate. If you do not create the correct types of indexes for a column, Sybase IQ can still resolve queries involving the column, but response may be slower than it would be with the correct index types.

If multiple indexes are defined on a particular column, Sybase IQ builds all the indexes for that column from the same input data.

Index Guidance from the Optimizer

If you set the **INDEX_ADVISOR** option on your database, Sybase IQ issues messages in the message log or query plan to suggest additional indexes that might improve performance. Messages focus on the following areas:

- Local predicate columns
- Single-column join key columns
- Correlated subquery columns
- Grouping columns

If you decide to follow the recommendations, you create the indexes yourself.

Adding and Dropping Indexes

If you discover later that you need additional indexes, it is simple to add them; however, it is usually faster to create all necessary indexes before you insert any data.

You can only rename or alter an index in a base table or global temporary table with the owner type USER. See the **ALTER INDEX** statement in *Reference: Statements and Options* for more information on renaming indexes and changing foreign key role names.

You can drop any optional index if you decide that you do not need it. See the **DROP INDEX** command in the **DROP** statement in *Reference: Statements and Options* for more information on dropping indexes.

Note: 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.

Note that ALTER TABLE DROP FOREIGN KEY CONSTRAINT does not remove the automatically-created non-unique **HG** index. You cannot drop a primary key if associated foreign keys remain. To remove such an index, drop it explicitly after issuing the ALTER TABLE DROP FOREIGN KEY command.

Benefits Over Traditional Indexes

Sybase IQ indexes offer the following benefits over traditional indexing techniques:

- 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.

Tools for Creating Sybase IQ Indexes

You can create a column index explicitly using either of two tools.

- Interactive SQL

  To create a Sybase IQ column index, use the SQL syntax in *Reference: Statements and Options > SQL Statements > CREATE INDEX statement*.

To create a Sybase IQ column index, use the SQL syntax in *Reference: Statements and Options > SQL Statements > CREATE INDEX statement*.

- Sybase Central

  To create a column index using Sybase Central, see "Creating Column Indexes" in *Introduction to Sybase IQ > Indexing and Loading Data*.

Interactive SQL Index Creation

If you do not specify an *index-type*, Sybase IQ creates an **HG** index.

Several front-end tools create an **HG** index automatically for this reason.

Index Creation Examples

The following two examples demonstrate Index Creation functionality.

The first example creates a High_Non_Group (**HNG**) index called `ShipIx` on the `ShipDate` column of the `SalesOrderItems` table.

```
CREATE HNG INDEX ShipIx
  ON dbo.SalesOrderItems (ShipDate)
```

The second example creates a Low_Fast (**LF**) index called `SalesOrderRegionIX` on the `Region` column of the `SalesOrder` table.

```
CREATE LF INDEX SalesOrderRegionIx
  ON dbo.SalesOrder (Region)
```

See also

- *The Compare (CMP) Index Type* on page 213

Status Messages for Index Loading

By default, after every 100,000 records are inserted and loaded into indexes, you receive a progress message.

To change the number of records, specify the **NOTIFY** option of **CREATE INDEX**, or the option **NOTIFY_MODULUS**. To prevent these messages, specify **NOTIFY 0**.

How to Execute 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.

These keywords can only be used when creating indexes on IQ tables, not temporary tables or SQL Anywhere tables. If one of these **CREATE INDEX** statements fails, they all roll back. See *Reference: Statements and Options*.

Concurrent Column Index Creation

In some cases, you can create more than one column index at the same time.

- 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 other gets an error saying that only 1 writer is allowed.
- If two connections issue **CREATE INDEX** statements on different tables, both proceed in parallel.
- If two connections issue **CREATE INDEX** statements on different tables but both tables participate in the same join index, then only one **CREATE INDEX** works; the other gets an error saying that only 1 writer is allowed.

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, the query results users want from it, and whether the data is used in ad hoc joins or join indexes.

For details of index types, and criteria to use for choosing the correct types, see the sections that follow.

Number of Unique Values in the Index

Sybase IQ indexes are optimized according to the number of unique (distinct) values they include.

When the number of distinct values reaches certain levels, choose indexes according to the recommendations in this table.

Table 12. Consideration Order

Number of Unique Values	Recommended Index Type
Below 1,000	**LF** (**HG** if table has <25,000 rows)
1000 and over	**HG** and/or **HNG**

Columns created when `MINIMIZE_STORAGE` option is ON, or for which you specify `IQ UNIQUE 65536` or less, are automatically placed in a form of the default index that is optimized for reduced storage, and improved performance for certain types of queries.

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

Sybase recommends certain index types for particular types of queries.

You should know 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**, and/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: Sybase IQ can still resolve queries involving a column indexed with the wrong index type, although it may not do so as efficiently.

This table shows recommended index types based on the 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. You should also consider the number of unique values and disk space. See the other tables in this section.

Table 13. Query type/index

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	**HNG, LF, HG**, Default
As **MIN/MAX** argument	**LF, HG, HNG**
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**,
Columns used in a join index	**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**, **HG**, or **HNG**; also **CMP**, **DATE**, **TIME**, **DTTM**
In **DATEPART** equality, range, and IN list predicates	**DATE**, **TIME**, **DTTM**
In a **CONTAINS** predicate	**WD**, **TEXT**

While **HNG** is recommended, in certain cases **LF** or **HG** is faster, and is often used in place of **HNG**. **HNG** tends to give consistent performance, while the performance of **LF** or **HG** with ranges depends on the size of the range selected.

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 a more important factor.

- With the set functions **COUNT**, **COUNT DISTINCT**, **SUM**, **MIN**, **MAX**, and **AVG**, in order to use any index other than the default, the entire query must be resolvable using a single table or join index.
- `BIT` data can only be used in the default index; `VARBINARY` data greater than 255 bytes can only be used in the default, **TEXT**, and **CMP** index types; `CHAR` and `VARCHAR` data greater than 255 bytes can only be used in the default, **CMP**, **TEXT**, and **WD** index types; **LONG VARCHAR** data can only be used 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.

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 14. 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 2 times larger	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 2 times larger	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 takes.

Because **CMP** indexes are always an additional index, they do not save disk space.

See also

- *Overlapping Versions and Deletions* on page 387

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. See the following table for a list of other indexes supported for each data type.

Table 15. 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
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

Data type	Supported indexes	Unsupported indexes
long varchar	WD, TEXT	CMP, HG, HNG, LF, DATE, TIME, DTTM
binary	CMP, HG, LF, TEXT	HNG, 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

Combining Index Types

When a column will be used in more than one type of query, more than one column index type might be appropriate.

These index types make good combinations.

Table 16. 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 = A reasonable combination

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.

Sybase IQ Index Types

This section explores in depth the reasons you might use each of the column index types.

The Fast Projection (FP) Default Index Type

When you create a permanent table in a Sybase IQ database, IQ stores all column values in a default index. This default index, called an **FP** (Fast Projection) index, optimizes projections and enables certain kinds of search conditions to be evaluated.

Each column has one **FP** index, and each **FP** is an array of *n* fixed-length entries where *n* is the number of rows in the table. Each column value is stored sequentially in ascending RecordID order.

With a small number of distinct or unique values, such as a state, date, or month field, you can create an optimized form of the **FP** that reduces the number of disk pages required, dramatically reducing both the storage required for a column and I/O costs for projection.

These optimized **FP** indexes have two pieces; a lookup table, where each distinct value in the column appears exactly once; and the logical array of values, where each element of the logical array is a key pointing to the location where the cell value is stored in the lookup table.

The **sp_iqindexmetadata** stored procedure generates a report describing a specified index or indexes belonging to a specified owner or table. **sp_iqindexmetadata** output shows whether a given index is a 1-byte, 2-byte, 3-byte, or flat style **FP** index. For details, see "sp_iqindexmetadata Procedure" in *Reference: Building Blocks, Tables, and Procedures*.

Tokenization can be applied to columns that have more than 64K distinct values.

FP(1) Index

When there are fewer than 256 column values, a 1-byte **FP** index is created.

The actual key value of each row is stored in the lookup table and the **FP** entry (lookup key) contains the index into the lookup table. For a 1-byte **FP** index, each **FP** entry occupies 1 byte.

With default main cache setting of 32MB and a default `FP_LOOKUP_SIZE_PPM` setting of 2500, adding a new distinct value using **LOAD**, **INSERT**, or **UPDATE** to a FP(1) column of char(2), binary(2), or small integer, converts FP(1) to flat **FP**.

FP(2) Index

When the number of column values is greater than 256, but is fewer than or equal to 65536, the lookup table grows. Each **FP** entry (lookup key) contains the index into the lookup table but occupies 2 bytes instead of 1.

To avoid the overhead of converting 1-byte entries into 2-byte entries, users can specify the `IQ UNIQUE` value to be greater than 256 and less than or equal to 65536 at table creation time.

FP(3) Index

When the distinct count of column values exceeds 65536, you can create a 3-byte **FP** index.

The FP(3) index is structurally similar to FP(1) and FP(2) indexes, with the following key differences:

- The maximum size of the FP(3) lookup table is 16777216, not 65536.
- The FP(3) index buffer storage contains lookup keys of 3 bytes each.

The 3-byte index stores values in a column (column data), as long as the distinct count does not exceed 16777216. Users can create a 3-byte index on columns only if the size of column data is greater than 3 bytes. Therefore, you cannot create an FP(3) index on columns with data types `BIT`, `TINYINT`, `SMALLINT`, `CHAR`(<=3), `VARCHAR`(<=3), `BINARY`(<=3) and `VARBINARY`(<=3). Sybase IQ also does not support FP(3) indexes for `LONG VARCHAR` and `LONG BINARY` data types.

To create an FP(3) index, either the `MINIMIZE_STORAGE` must be set ON, or the column must have been created with an `IQ UNIQUE` constraint value between 65537 and 16777216, including these two values. An FP(3) may also be created from an FP(2) index, once the unique count exceeds 65536, but only for data sizes shown in the *Sybase IQ fast projection (FP) indexes* table below and excluding data types `BIT`, `TINYINT`, `SMALLINT`, `CHAR`(<=3), `VARCHAR`(<=3), `BINARY`(<=3) and `VARBINARY`(<=3). See the **MINIMIZE_STORAGE** option in *Reference: Statements and Options*.

Behavior Changes

There are some differences in the behavior of **FP** indexes from versions of Sybase IQ versions earlier than 15.0.

Table 17. Sybase IQ fast projection (FP) indexes

Distinct count	Column data size = 1 byte	Column data size = 2 bytes	Column data size = 3 bytes	Column data size >3 bytes
<257	FP(1)	FP(1)	FP(1)	FP(1)
257 - 65536	—	FP(2)	FP(2)	FP(2)
65537-16777216	—	—	Flat	FP(3)
>16777216	—	—	—	Flat

Like FP(1) and FP(2) indexes, the FP(3) index is not supported for columns whose data type is wider than 255 bytes or less than 4 bytes wide.

The creation of an FP(3) index, overflow, or a forced transition to an FP(3) index is permitted only if the space used by the lookup table is less than the current value of the `FP_LOOKUP_SIZE` option and less than the portion of the main cache specified by the current setting of `FP_LOOKUP_SIZE_PPM`.

The maximum number of lookup pages used in Sybase IQ is controlled by the `FP_LOOKUP_SIZE` option and the `FP_LOOKUP_SIZE_PPM` option, whichever value is smaller. See *Reference: Statements and Options*.

The following table calculates the maximum number of distinct values that can be supported in an FP(3) index based on the following formula:

```
FP_LOOKUP_SIZE / (Column-Data size + Cardinality size)
```

Cardinality size is the space reserved to store cardinality of all individual data in the lookup store. Cardinality size can have a value of either 4 or 8 bytes. In this example, it has a maximum value of 8 bytes.

This table also contains examples, with values less than 16777216, where a rollover to a flat **FP** occurs for a smaller unique count than the expected 16777216.

Table 18. Maximum unique values in FP(3)

	Column data type width (bytes)					
FP_LOOKUP_SIZE (MB)	4	8	32	64	128	255
1 MB	87381	65536	26214	14563	7710	3986
4 MB	349525	262144	104857	58254	30840	15947
8 MB	699050	524288	209715	116508	61680	31895
16 MB	1398101	1048576	419430	233016	123361	63791
32 MB	2796202	2097152	838860	466033	246723	127583
64 MB	5592405	4194304	1677721	932067	493447	255166
128 MB	11184810	8388608	3355443	1864135	986895	510333
256 MB	16777216	16777216	6710886	3728270	1973790	1020667

Note: The values illustrated in this table are estimates for the number of unique values in a column for the given value of option `FP_LOOKUP_SIZE`; actual values may vary. Such variations are possible because counts can be stored as 4 bytes or 8 bytes.

This table is based on the condition that the value of `FP_LOOKUP_SIZE` is less than or equal to the value of `FP_LOOKUP_SIZE_PPM`.

See also

- *UNIQUE Constraints on Columns or Tables* on page 351
- *Configuring FP(3) Indexes* on page 208

Configuring FP(3) Indexes

You may need to adjust the temporary cache size when configuring 3-byte indexes.

You can set values using the server startup command line parameter **-iqtc** or using the **sa_server_option** system procedure **temp_cache_memory_mb** option as follows:

```
CALL sa_server_option('temp_cache_memory_mb', value)
```

The enumerated FP indexes use a hash object to manage the values represented in the column. The size of the hash object used with a 3-byte **FP** can get large, depending on the number of distinct values and the width of the column. With a large enough temporary cache allocation, increasing the value of the option `HASH_PINNABLE_CACHE_PERCENT` to more than the default value of 20 percent can improve performance by allowing the entire hash object to remain in the cache.

FP(3) Cache Usage

To maximize the use of FP(3) indexes, set the `FP_LOOKUP_SIZE` option to a value larger than the default of 16MB.

FP(3) Loads

Columns with a 3-byte index require additional cache to load data. Set `FP_LOOKUP_SIZE` to an appropriate value before loading columns with 3-bytes indexes.

If a scarcity of pinned buffers occurs, Sybase IQ returns a warning in the `.iqmsg` file, which also contains notification of possible thrashing:

```
Warning: Hash Insert forced buffer unpinning detected for FP Index
Warning: Hash Insert thrashing detected for FP Index
```

See also

- *FP(3) Index* on page 206
- *UNIQUE Constraints on Columns or Tables* on page 351

Flat FP Index

When the number of distinct values exceeds 16777216, no lookup table is created. Each flat style FP index contains an actual column cell value.

If `MINIMIZE_STORAGE` is ON, you can avoid the overhead of converting lookup FP entries into flat style. When the distinct row count for a particular field increases to more than 16777216, an FP(3) index is automatically converted to a flat style FP index. To create a flat FP, specify the `IQ UNIQUE` value to be greater than 16777216 at table creation time.

Note: When you create a table with the `DATE` data type, a 2-byte FP index is created on the `DATE` field, which is independent of the settings in database option `MINIMIZE_STORAGE`.

To create a 3-byte FP or flat style FP index on the `DATE` field, use these values when creating the table:

- For a 3-byte FP index— `IQ UNIQUE` value between 65537 and 16777216.
- For a flat style FP index— `IQ UNIQUE` value greater than 16777216.

The Low_Fast (LF) Index Type

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 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

The recommended use for an LF index is as follows:

Use an **LF** index when:

- A column has fewer than 1,000 unique values.
- 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

The following table lists advantages and disadvantages of Low_Fast indexes.

Table 19. LF advantages/disadvantages

Advantages	*Disadvantages*
This index is fast, especially for single table **SUM**, **AVG**, **COUNT**, **COUNT DISTINCT**, **MIN**, and **MAX** operations.	Can only be used for a maximum of 10,000 unique values. Cannot use this index 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 also 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. Only when the number of unique values is high should other indexes (**HG** and **HNG**) be considered. For range queries with a high number of unique values, also consider having an **HNG** index.

The High_Group (HG) Index Type

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

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 reference to multiple columns. This change is transparent to users, but improves query performance.

Note: Foreign key columns require their own, individual **HG** index. However, if a join index exists, the same column cannot have both an explicitly created HG index and a foreign key constraint.

Advantages and Disadvantages of High_Group

High_Group indexes have these advantages and disadvantages.

Table 20. HG advantages/disadvantages

Advantages	***Disadvantages***
Quickly processes queries with **GROUP BY**. This index facilitates join index processing. It is one of indexes recommended for columns used in join relationships. **LF** is the other.	This index needs additional disk space compared to the **HNG** index (it can take up as much as three times more space than raw data). This index type takes the longest time to populate with data, and to delete. 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

The following is a comparison of other 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

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

Sybase IQ creates a High_Group index by default whenever you issue a **CREATE INDEX** statement without specifying an index type.

Sybase IQ automatically creates a High_Group index for any **UNIQUE**, **FOREIGN KEY**, or **PRIMARY KEY** constraint. For foreign keys of a single column, 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.

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, see "Multicolumn Indexes" in *Reference: Statements and Options > SQL Statements . CREATE INDEX Statement > Notes*.

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.

The High_Non_Group (HNG) Index Type

Add an **HNG** index when you need to do range searches.

An **HNG** index requires approximately three times less disk space than an **HG** index requires. On that basis alone, if you do not need to do group operations, use an **HNG** index instead of a **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 and/or you want to enforce uniqueness, use a **HG** index.

Note: Using the **HNG** index in place of a **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

Be aware of the recommended usage of the **HNG** index.

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

See the following table for advantages and disadvantages of using a High_Non_Group index.

Table 21. 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 few exceptions described below.	This index is not recommended for **GROUP BY** queries. Index not possible if uniqueness enforced. Cannot use this index 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 can't 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

The High_Group index is also appropriate for an **HNG** column.

The Compare (CMP) Index Type

A Compare (CMP) index is an index on 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.

See also

- *Index Creation Examples* on page 199

Recommended Use

Recommended usage for the CMP Index Type.

The **CMP** index can be created on columns that are NULL, NOT NULL, or a mixture. The **CMP** index cannot be unique. Note that 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, the following commands show how to create a table, then create appropriate Compare 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)
```

The following 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

The following restrictions apply to **CMP**:

- You can drop **CMP** indexes.
- **CMP** indexes cannot be unique.
- **CMP** indexes are not replicated in underlying join indexes.
- An exception is raised if you attempt to alter or delete a column that is defined in a **CMP** index.
- Users 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.

The Containment (WD) Index Type

The Containment (**WD**) index allows you to store words from a column string of `CHAR`, `VARCHAR`, and `LONG VARCHAR` data.

Note: In order to create `LONG VARCHAR` columns, you must be specifically licensed to use the unstructured data analytics functionality. For details on the Unstructured Data Analytics Option, see *Unstructured Data Analytics in Sybase IQ*.

Recommended Use

Use a **WD** index for the fastest access to columns that contain a list of keywords (for example, in bibliographic record or Web page).

The following restrictions apply to **WD**:

- You cannot specify the UNIQUE attribute.
- The **WD** index is 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. The Sybase Central Create Index wizard does not indicate this when it prompts for delimiter characters, and returns an error if you omit them.
- If the DELIMITED BY clause is omitted or the *separators-string* value specified is the empty string (single quotes), then 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 for the hyphen and the single quotation mark, which are part of words by default. There are 64 separators in the default separator set.
- If multiple DELIMITED BY and LIMIT clauses are specified, 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, Sybase recommends that you specify a space as at least one of the separators or use the default separator set. 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

See the following table for advantages and disadvantages of using a **WD** index.

Table 22. WD advantages/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 only use this index if data in your columns is `CHAR`, `VARCHAR`, or `LONG VARCHAR`.

The Date (DATE), Time (TIME), and Datetime (DTTM) Index Types

Three index types are used to process queries involving date, time, or datetime quantities:

- A **DATE** index is used on columns of data type `DATE` to process certain queries involving date quantities.
- The **TIME** index is used on columns of data type `TIME` to process certain queries involving time 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, TIME, and DTTM Index Types

Use a **DATE**, **TIME**, or **DTTM** index in the following cases, when the `DATE`, `TIME`, `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`, `TIME`, `DATETIME`, or `TIMESTAMP` column, **LF** and **HG** indexes have the best performance. If an **LF** or **HG** index is not available, then the **DATE**, **TIME**, or **DTTM** index is used to get the result.

If a `DATE`, `TIME`, `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 do fast equality.

The table `tab` used in the examples in this section contains columns defined as follows:

```
CREATE TABLE tab
(col1 DATE,
```

```
col2 DATETIME,
col3 TIME);
```

See also

- *Additional Indexes* on page 220

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 the following 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, whose parameter 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, whose parameter 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, whose parameter 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/TIME/DTTM

See the following table for advantages and disadvantages of using a **DATE**, **TIME**, or **DTTM** index.

Table 23. DATE/TIME/DTTM advantages/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**, **TIME**, 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`, `TIME`, `DATETIME`, or `TIMESTAMP` data type.

Restrictions on DATE/TIME/DTTM Indexes

The following restrictions currently apply to **DATE**, **TIME**, 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 faster (up to twice as fast) than **HNG** in the supported cases. In the special cases discussed in the "Recommended use" section, the performance of the **DATE**, **TIME**, and **DTTM** indexes is even better. 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

It is recommended to always have 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 less than 1000 distinct values (i.e., less than three years of dates).

See also

- *Recommended Use of DATE, TIME, and DTTM Index Types* on page 216

TEXT Indexes

o use **TEXT** indexes, you must be specifically licensed to use the unstructured data analytics functionality.

Note: T See *Unstructured Data Analytics in Sybase IQ*.

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

Create a TEXT Index Using Sybase Central.

See *Unstructured Data Analytics in Sybase IQ*.

Optimizing Performance for Ad Hoc Joins

Indexes can be created to optimize ad hoc join performance.

To gain the fastest processing of ad hoc joins, create a Low_Fast or High_Group index on all columns that may be referenced in:

- **WHERE** clauses of ad hoc join queries
- **HAVING** clause conditions of ad hoc 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.

Selecting an Index

Certain criteria, such as the number of unique values, help you to identify appropriate index types for your 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**

Criteria to identify	Index to select
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.	

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 and as part of any join indexes that include the column. **CMP** and multicolumn **HG** indexes are the only exception to this rule.

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. Data is also inserted and *synchronized* automatically when you add an index to previously loaded tables that are part of a join index.

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.

See also

• *Synchronizing Join Indexes* on page 229

Using Join Indexes

If you know that certain tables in the same database will typically be joined in a consistent way, you may want to create a *join index* for those tables.

When you create a join index, Sybase IQ produces a new internal structure that relates table columns. It represents two or more tables, including the inner, left outer, and right outer rows.

See also

• *Types of Tables* on page 180
• *Rules and Checks for Valid Data* on page 341

Join Indexes Improve Query Performance

Join indexes usually provide better query performance than when table joins are first defined at query time (ad hoc joins). In many situations, however, you can gain optimal performance on joined columns without creating join indexes.

Loading Considerations for Join Indexes

Join indexes require more space and time to load than other IQ indexes. To load a join index, you must first load the underlying tables, and then load the join index.

How Join Indexes are Used for Queries

After you create a join index, its use is determined by the criteria of the **SELECT** statement.

If a join index exists that joins the tables in the **FROM** clause by the relationship specified in the **WHERE** clause, or if a join index exists that is based on ANSI join syntax for natural or key joins, the join index is used to speed up queries. Otherwise, ad hoc joins between indexes on the individual tables are performed at query time. If there is a join index for a subset of tables in the **SELECT**, Sybase IQ uses it to speed up the resulting ad hoc join.

Relationships in Join Indexes

Sybase IQ join indexes support one-to-many join relationships. A simple example of a one-to-many relationship is a sales representative to a customer. A sales representative can have more than one customer, but a customer has only one sales representative.

There can be multiple levels of such relationships. However, you always specify join relationships between two tables, or between a table and a lower level join. The table that represents the "many" side of the relationship is called the *top table*.

See also

- *Join Hierarchy Overview* on page 224
- *The Join Hierarchy in Query Resolution* on page 225
- *Steps in Creating a Join Index* on page 228

When a Join Becomes Ad Hoc

If there is no join index that handles all of the reference tables involved in a query, the query is resolved with an ad hoc join

. Because you cannot create a join index to represent a many-to-many join relationship, you can only issue ad hoc queries against such a relationship. Ad hoc queries provide flexibility, but in some situations this flexibility comes at the expense of performance. If you have sufficient space for the join indexes, and you do not require many-to-many relationships or multilevel star join indexes, you may find it helpful to create join indexes where performance is critical.

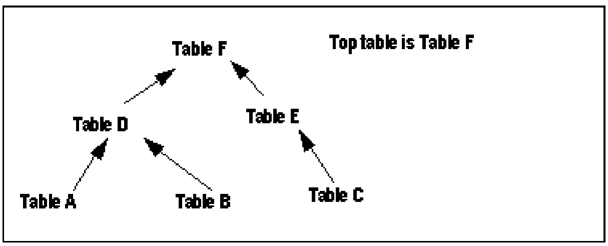
Join Hierarchy Overview

All join relationships supported by Sybase IQ must have a hierarchy. Think of a join hierarchy as a tree that illustrates how all the tables in the join are connected.

Sybase IQ join hierarchies have one table at the top of the tree where the join ends. This table, known as the *top table*, does not connect to any other tables, although other tables connect to it. The top table always represents the "many" side in a one-to-many relationship.

Depending on the complexity of the join, there could be a straight line of tables down to the bottom of the tree and the beginning of the join, or there could be many branches off to the side as you move down the tree. The following figure shows a join hierarchy with two branches.

Figure 1: Hierarchy of a join relationship



In a join hierarchy:

- A table can occur only once
- A table can only connect out once (one arrow leaving it)
- All tables must be connected

See also

- *Relationships in Join Indexes* on page 223
- *The Join Hierarchy in Query Resolution* on page 225
- *Steps in Creating a Join Index* on page 228
- *Examples of Join Relationships in Table Definitions* on page 230
- *Specifying the Join Type When Creating a Join Index* on page 231
- *Specifying Relationships When Creating a Join Index* on page 232
- *Types of Join Hierarchies* on page 234

Columns in the Join Index

Suppose that you join Tables A through E in a join index called `ABCDE`. If each table has two columns of data, expect the join index to have a total of fourteen columns.

Sybase IQ creates an additional column, the `ROWID` column, for each of the joined tables except the top table. In this case, there are ten columns (two from each of the five tables), plus four `ROWID` columns.

You can use the **NOTIFY** option of the **LOAD TABLE** or **INSERT** statement to receive notification messages when you insert into a column index. These messages identify each column in a join index, including the `ROWID` column.

You can set the frequency of these messages with the **NOTIFY_MODULUS** option, and override the option value in either the **CREATE DATABASE** or **LOAD TABLE** command.

Join index columns must have identical data type, precision, and scale.

See also

- *Interpreting Notification Messages* on page 515
- *Load and Insert Messages* on page 259
- *Message Log Wrapping* on page 11
- *Version Information in Message Logs* on page 9

The Join Hierarchy in Query Resolution

Sybase IQ can use the same join index to resolve a query that involves the full join relationship specified in the join index, or a query that involves any contiguous subset of that relationship. You do not have to create separate join indexes for the subset relationships.

For example, assume that join index `ABCDEF` joins the tables illustrated in the *Hierarchy of a join relationship* figure. Sybase IQ can use join index `ABCDEF` to resolve any queries that involve:

- The entire relationship
- Table `A` to Table `D`

- Table `A` to Table `D` to Table `F`
- Table `B` to Table `D`
- Table `B` to Table `D` to Table `F`
- Table `D` to Table `F`
- Table `C` to Table `E`
- Table `E` to Table `F`
- Table `C` to Table `E` to Table `F`

However, Sybase IQ cannot use join index `ABCDEF` to resolve queries against, for example, Table `E` to Table `D`.

See also

- *Join Hierarchy Overview* on page 224
- *Relationships in Join Indexes* on page 223
- *Steps in Creating a Join Index* on page 228

One-to-Many Relationship

In a one-to-many join relationship, one row in one table potentially matches with one or more rows in another table, and there is not more than one row in the first table that matches with the same row(s) in the second table.

For this to be true, the values in the join column in the first table must be unique.

It is possible that either table has no match on the other table. This constitutes an outer join. Sybase IQ fully supports outer joins.

If the join column is made up of more than one column, the combination of the values must be unique on the "one" side. For example, in the `iqdemo` database, the `ID` in the `Customers` table and the `CustomerID` in the `SalesOrders` table each contain a customer ID. The `Customers` table contains one row for each customer and, therefore, has a unique value in the `ID` column in each row. The `SalesOrders` table contains one row for each transaction a customer has made. It is likely that there are many transactions for each customer, so there are multiple rows in the `SalesOrders` table with the same value in the `CustomerID` column.

If you join `Customers.ID` to `SalesOrders.CustomerID`, the join relationship is one-to-many. As you can see in the following example, for every row in `Customers`, there are potentially many matching rows in `SalesOrders`. (Output for example has been limited to 15 rows.)

```
SELECT SalesOrders.ID, SalesOrders.CustomerID,
Customers.GivenName
from SalesOrders, Customers
where SalesOrders.CustomerID = Customers.ID
```

```
ID   CustomerID    GivenName
---- ----------    ---------
2001        101    Michaels
```

```
2005        101    Michaels
2125        101    Michaels
2206        101    Michaels
2279        101    Michaels
2295        101    Michaels
2337        101    Michaels
2389        101    Michaels
2447        101    Michaels
2560        101    Michaels
2583        101    Michaels
2002        101    Beth
2142        101    Beth
2318        101    Beth
2338        101    Beth
```

Warning! If the one-to-many relationship is incorrect, the join cannot be synchronized until you remove the extra rows from the "one" table. If you try to synchronize, you get a Duplicate Row error, and the transaction rolls back.

When you create a join index, you use ANSI FULL OUTER join syntax. Sybase IQ stores the index as a full outer join. Later, when you issue queries against the columns in a join index, you can specify inner, left outer, and right outer join relationships as well as full outer joins. Sybase IQ uses only the parts of the join index needed for a given query.

Multiple Table Joins and Performance

Rules for multiple table joins are:

- A table can be on the "one" side of a one-to-many relationship just once. For example, you cannot have a join index or a join query where Table A is joined to Table B in a one-to-many relationship, and Table A is joined to Table C in a one-to-many relationship. You need to create separate join indexes for each of these relationships.
- A table can appear in the relationship hierarchy only once. So, for example, you cannot predefine a join relationship query where Table A is joined to Table B, Table B is joined to Table C, and Table C is joined to Table A. You can use predefined joins to query on the Table A to Table B and the Table C to Table A relationships separately. To do so, create a separate join index for each of these relationships.
- A table can be joined to another table, or to a join definition. For example, you can create a join index that joins Table A to Table B, or a join index that joins Table C to the join of Tables A and B.
- The top table in the hierarchy is the "many" side of a one-to-many relationship with the rest of the hierarchy.
- The most useful join indexes are usually two-table joins.

In some circumstances, you may want to create a separate join index for a subset of the join relationship. If the top table in the subset of the join index has a significantly smaller number of rows than the top table in the full join index, a query on the subset may be faster than the same query on the full join index if only tables in the subset are used in the query.

Of course, this approach requires more disk space to build an additional join index and more index building time (not to mention increased maintenance). In the case of a subset join index, the additional join index repeats a subset of the information already in the full join index. You must decide whether the query speed or disk space usage of your application is more important for this particular join relationship. Keep in mind also that in the current version of Sybase IQ, join indexes may not provide the same performance advantage as in previous releases, especially when the relationship hierarchy includes multiple levels.

Steps in Creating a Join Index

In order to create a join index you must perform all of the following steps.

1. Create the tables involved in the join index, using the **CREATE TABLE** command, or using Sybase Central. These must be permanent tables; you cannot use a temporary table to create a join index.
2. Identify the *join condition* that relates specific pairs of columns in the underlying tables involved in any one join.

   It is important to define a schema for your database, to clarify join conditions and other assumptions about the structure of your data. The schema should represent foreign key-primary key relationships, and follow other best practices of schema design. Columns related by foreign key must have matching data types, precision, and scale.

   Where the relationship is based on a key join, you must define join conditions as referential integrity constraints—primary and foreign key declarations—in the **CREATE TABLE** commands in step 1 or in **ALTER TABLE** commands.
3. Create a primary key for each column involved in a join.
4. Create column indexes for the tables being joined.

   When Sybase IQ creates a join index between tables, the IQ column index types and data types already defined on the single tables are used in the join index. Multicolumn indexes on base tables are not replicated in join indexes created using those base tables.
5. Load the data into the tables, using the **LOAD TABLE** command. You also can add data to existing tables using the **INSERT INTO** command.

   Note: You must insert into each table in the join index as a single-table insert, rather than into the join index itself. This approach conforms to ANSI rules for indexed data.
6. Create the join index by issuing the **CREATE JOIN INDEX** command, or in Sybase Central with the Create Join Index wizard. You specify the join hierarchy as part of this step.
7. Depending on the order in which you perform these steps, you may need to synchronize the tables in the join index, as described below. If data exists in the join tables, synchronization occurs automatically.

The index remains unavailable until all steps are complete. However, you can adjust the order of some steps, depending on the needs of your site:

- You can combine steps 1 and 2 by defining relationships when you create the table.
- You can load the data either before or after you create the join index. If you load the data into the underlying column indexes after you create the join index, you must perform the synchronization step.

See also

- *Join Hierarchy Overview* on page 224
- *Relationships in Join Indexes* on page 223
- *The Join Hierarchy in Query Resolution* on page 225

Privileges Needed to Create a Join Index

You must be the owner of a table or the DBA to create, alter, or synchronize a join index that includes that table. If you are not the DBA, you need to be the owner of the table and have RESOURCE authority in order to create a join index.

For complete syntax, see **CREATE TABLE** statement, **ALTER TABLE** statement, **LOAD TABLE** statement, **INSERT** statement, and **SYNCHRONIZE JOIN INDEX** statement in *Reference: Statements and Options*. The sections that follow explain more about creating a join index.

See also

- *Data Import and Export* on page 241

Synchronizing Join Indexes

The data in join index tables must be synchronized before you can use a join index. Synchronization ensures that the data is loaded in the correct order for the joins.

Synchronization occurs automatically when you create the join index. Synchronizing before completing the transaction that loads or inserts data also makes tables available immediately for all readers. Once data is loaded, however, you must synchronize the join index explicitly, with one exception: the join index is synchronized automatically when changes are made to the top table of the join hierarchy.

To synchronize explicitly, issue the following command:

```
SYNCHRONIZE JOIN INDEX [join-index-name [, join-index-name]
```

If you omit the index names, Sybase IQ synchronizes all join indexes.

See also

- *Adding Column Indexes After Inserting Data* on page 222

Performance Tips for Synchronization

Synchronization can be time consuming.

To improve performance, try these suggestions:

- Schedule synchronization during off-peak hours.
- Synchronize join indexes individually rather than all at once.
- Synchronize after executing an entire set of insertions and deletions. It is not a good idea to synchronize after every insertion or deletion, as the time it takes to update a join index depends significantly on the order of the updates to the tables. Synchronizing sets of

updates allows Sybase IQ to pick the optimal order for applying the table changes to the join index.

Defining Join Relationships Between Tables

When you create a join index, you must specify the relationship between each related pair in the join. A related pair is always two tables, however, you can also specify a relationship by relating a table to another join relationship.

Depending on the relationship, you specify it either once or twice:

- *Key joins* relate the primary key of one table to a foreign key in another table. For key joins you must specify a **PRIMARY KEY** and **FOREIGN KEY** when you create or alter the underlying tables, using the **CREATE TABLE** or **ALTER TABLE** command.
- For all joins, you specify the relationship when you create the join index, using the **CREATE JOIN INDEX** command. The join is defined by the order in which you list the tables, by the columns you specify, and by the join type: key join, natural join, or ON clause join.

Rules for join relationships are:

- Each pair of tables in a join relationship must have at least one join column.
- The join column must exist in both tables.
- A pair of tables can have more than one join column, as long as they have the same number of columns and the join column holds the same position in each table list when you specify it. The order of the lists for the two tables determines how the columns are matched.

Using Foreign References

Sybase IQ uses foreign keys to define the relationships among columns that will be used in join indexes, and to optimize queries.

Note that key joins, which rely on foreign keys, are required for certain types of join indexes.

Sybase IQ does not support key join indexes based on multicolumn foreign keys.

Examples of Join Relationships in Table Definitions

The following example shows how to specify the join relationship by means of primary and foreign keys.

In this case, one customer can have many sales orders, so there is a one-to-many relationship between the `ID` column of the `Customers` table (its primary key) and the `CustomerID` column of the `SalesOrders` table. Therefore, designate `CustomerID` in `SalesOrders` as a **FOREIGN KEY** that references the `ID` column of the `Customers` table.

The first example creates the `Customers` table, with the column `ID` as its primary key. To simplify the example, other columns are represented here by ellipses (...).

```
CREATE TABLE GROUPO.Customers
( ID INTEGER NOT NULL,
```

```
...
PRIMARY KEY (ID),)
```

Next, create the `SalesOrders` table with six columns, specifying the column named `CustomerID` as the primary key. You must also add a foreign key relating the `CustomerID` column of the `SalesOrders` table to the `ID` column of the `Customers` table.

You can add the foreign key either when you create the table or later. This example adds the foreign key by including the **REFERENCES** clause as a column constraint in the **CREATE TABLE** statement.

```
CREATE TABLE GROUPO.MySalesOrders
(ID INTEGER NOT NULL,
CustomerID INTEGER
REFERENCES GROUPO.Customers(ID),
OrderDate DATE NOT NULL,
FinancialCodesID CHAR(2),
Region CHAR(7),
SalesRep INTEGER NOT NULL,
PRIMARY KEY (ID),)
```

Alternatively, you can create the table without the **REFERENCES** clause, and then add the foreign key later, as is done in the following **ALTER TABLE** statement. You may issue one or the other of these statements, but not both:

```
ALTER TABLE GROUPO.MySalesOrders
ADD FOREIGN KEY ky_so_customer (CustomerID)
REFERENCES GROUPO.Customers (ID)
```

See also

- *Join Hierarchy Overview* on page 224
- *Specifying the Join Type When Creating a Join Index* on page 231
- *Specifying Relationships When Creating a Join Index* on page 232
- *Types of Join Hierarchies* on page 234

Specifying the Join Type When Creating a Join Index

The join type is always **FULL OUTER**, the keyword **OUTER** being optional.

You also need to do one of the following:

- If you are joining equivalent columns with the same name from two tables, you specify that it is a **NATURAL JOIN**.
- If you are joining columns based on keys, you must also have specified the relationship in the underlying tables as a **FOREIGN KEY** that references a **PRIMARY KEY**.
- If you are joining equivalent values (an *equijoin*) in columns from two tables, you specify an **ON** clause.

These rules conform to ANSI syntax requirements.

See also

- *Examples of Join Relationships in Table Definitions* on page 230
- *Join Hierarchy Overview* on page 224
- *Specifying Relationships When Creating a Join Index* on page 232
- *Types of Join Hierarchies* on page 234

Specifying Relationships When Creating a Join Index

For non-key joins, the order in which you specify tables when you create the join index determines the hierarchy of the join relationship between the tables.

The **CREATE JOIN INDEX** statement supports two ways to specify the join hierarchy:

- List each table starting with the lowest one in the hierarchy, and spell out the join relationship between each pair of tables. The last table in the list will be the top table in the hierarchy. For example, in the *Hierarchy of a join relationship* figure, F is the top table, E is below it, and C is at the bottom of the hierarchy. You could specify the join hierarchy for these three tables as follows:

  ```
  C FULL OUTER JOIN E FULL OUTER JOIN F
  ```
- Use parentheses to control the order in which the join relationships are evaluated. Parentheses control evaluation order just as they do in mathematics, that is, innermost pairs are evaluated first. With this method you start with the top table in the outermost set of parentheses, then any intermediate levels, and include the lowest two levels in the innermost parentheses. Using this method, you would specify the same three tables as follows:

  ```
  (F FULL OUTER JOIN (C FULL OUTER JOIN E))
  ```

  Note that the lowest level table appears first in the innermost parentheses, just as it does in the first method.

Note: While you can join these three tables in the way described here, in order to create the complete hierarchy shown in the *Hierarchy of a join relationship* figure you would need to use key joins.

When you create a join index, a message in the log identifies the top table in the join. For example,

```
[20691]: Join Index 'join_on_tabletable' created from the following
join relations:
 [20694]:      Table Name             Relationship
 [20697]:
-----------------------------------------------------------------
 [20696]: 1. join_on_table_a joined to 'join_on_table_b' One >> Many
 [20692]: The ultimate/top table is join_on_table_b
 [20697]:
-----------------------------------------------------------------
```

See also

- *Examples of Join Relationships in Table Definitions* on page 230

- *Join Hierarchy Overview* on page 224
- *Specifying the Join Type When Creating a Join Index* on page 231
- *Types of Join Hierarchies* on page 234

Issuing the CREATE JOIN INDEX Statement

The following examples illustrate how to issue CREATE JOIN INDEX statements.

Example 1: Key Join

This example creates a join index for the key join between the `SalesOrders` table and the `Customers` table. This is a key join based on the foreign key `ky_so_customer`, which relates the `CustomerID` column of `SalesOrders` to the primary key `ID` of the `Customers` table. You can give the index any name you want. This example names it `ky_so_customer_join` to identify the foreign key on which the key join relies.

```
CREATE JOIN INDEX ky_so_customer_join
FOR GROUPO.Customers FULL OUTER JOIN GROUPO.SalesOrders
```

Example 2: ON Clause Join

This example creates a join index for the same two tables using an **ON** clause. You could use this syntax whether or not the foreign key existed.

```
CREATE JOIN INDEX customer_sales_order_join
FOR GROUPO.Customers FULL OUTER JOIN GROUPO.SalesOrders
ON Customers.ID=SalesOrders.CustomerID
```

Example 3: Natural Join

To create a natural join, the joined columns must have the same name. If you created a natural join on the tables in previous examples, you would not get the expected results at all. Instead of joining the `ID` column of `Customers` to the `CustomerID` column of `SalesOrders`, the following command would join the dissimilar `ID` columns of the two tables, which is not allowed:

```
CREATE JOIN INDEX customers_sales_order_join
FOR GROUPO.Customers NATURAL FULL OUTER JOIN GROUPO.SalesOrders
```

A natural join between the `id` columns of `SalesOrders` and `SalesOrderItems` makes more sense. In this case, the columns with the same name should contain matching values. The command to create a join index based on a natural join between these two tables is:

```
CREATE JOIN INDEX sales_order_so_items_join
FOR GROUPO.SalesOrders NATURAL FULL OUTER JOIN
GROUPO.SalesOrderItems
```

Creating a Join Index in Sybase Central

Follow these steps to create a join index in Sybase Central.

To create a join index in Sybase Central, see *Introduction to Sybase IQ > Indexing and Loading Data > Creating join indexes* .

Types of Join Hierarchies

Sybase IQ supports two different types of join hierarchies.

The supported hierarchies are:

- Linear joins
- Star joins

You create ad hoc joins for both linear and star joins. Join indexes are designed for use with linear joins.

See also

- *Examples of Join Relationships in Table Definitions* on page 230
- *Join Hierarchy Overview* on page 224
- *Specifying the Join Type When Creating a Join Index* on page 231
- *Specifying Relationships When Creating a Join Index* on page 232

Linear Joins

You can think of a linear join as a tree with no branches.

Each table in the hierarchy is related to the table above it, until you reach the top table. Tables A, D, and F constitute a linear join hierarchy. Tables C, E, and F form another linear join hierarchy.

In a linear join, each pair of tables represents a one-to-many relationship, in which the lower table of the pair is the “one” side, and the higher table of the pair is the “many” side. Linear join hierarchies can rely on any of the underlying join conditions: key join, natural join, or ON clause join.

Star Joins

You can picture a star join as a structure with many branches, in which each branch is directly related to one table in the middle.

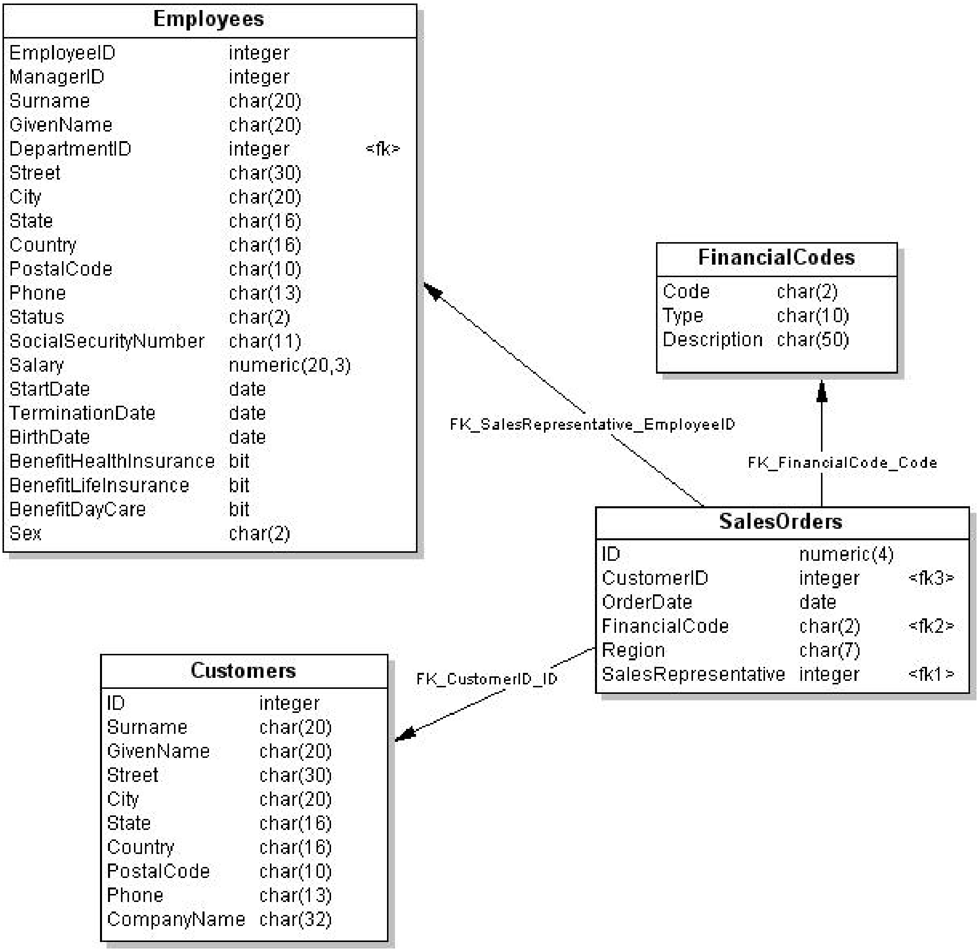
In the *Hierarchy of a join relationship* figure, Tables D, F, and E form a very simple star join. More commonly, Table F would be at the center of many tables, each of which is joined to Table F.

In a star join, multiple tables are related to one table at the center of the join, in a one-to-many relationship. The one table at the center of the join represents the “many” side of the relationship, while each of the tables around it represent the “one” side of the relationship. Each table on the “one” side holds a set of values with its own unique primary key. A foreign key in the table on the “many” side of the relationship relates that table to the primary key of the table on the “one” side of the relationship.

The “many” table at the center of the star is sometimes called the *fact* table. The “one” tables related to it are called the *dimension* tables.

Example

In the following example, the `SalesOrders` table contains three foreign keys, each of which is related to the primary key of another table.



You can create this table using the following commands:

```
CREATE TABLE GROUPO.SalesOrders (
   ID NUMERIC (4) NOT NULL IQ UNIQUE (648),
   CustomerID INTEGER NOT NULL IQ UNIQUE (109),
   OrderDate date NOT NULL IQ UNIQUE (376),
   FinancialCode CHAR (2) NULL IQ UNIQUE (1),
   Region CHAR (7) NULL IQ UNIQUE (5),
   SalesRepresentative  INTEGER
   NOT NULL IQ UNIQUE (11)
);

COMMENT ON TABLE GROUPO.SalesOrders is
```

```
'sales orders that customers have submitted
to the sporting goods company';

ALTER TABLE GROUPO.SalesOrders
ADD FOREIGN KEY FK_CustomerID_ID (CustomerID)
REFERENCES GROUPO.Customers (ID)
ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE GROUPO.SalesOrders
ADD FOREIGN KEY FK_FinancialCode_Code (FinancialCode)
REFERENCES GROUPO.FinancialCodes (Code)
ON DELETE RESTRICT ON UPDATE RESTRICT;

ALTER TABLE GROUPO.SalesOrders
ADD FOREIGN KEY
FK_SalesRepresentative_EmployeeID
(SalesRepresentative)
REFERENCES GROUPO.Employees (EmployeeID)
ON DELETE RESTRICT ON UPDATE RESTRICT;
```

As shown in the figure, the `Salesorders` table is at the center of the star join. Each of its foreign key columns can contain many instances of the primary key it refers to. For example, if you enter:

```
SELECT SalesRepresentative FROM SalesOrders
WHERE SalesRepresentative = 299
```

the results show 114 rows with 299 in the `SalesRepresentative` column.

However, if you enter:

```
SELECT EmployeeID FROM Employees
WHERE EmployeeID = 299
```

the results show only one row with 299 in the `EmployeeID` column.

Note: Query optimizations for all joins rely heavily on underlying primary keys. They do not require foreign keys. However, you can benefit from using foreign keys. Sybase IQ enforces foreign keys if you set up your loads to check for primary key-foreign key relationships.

Sybase IQ does not support star-join style join indexes that use multiple join key columns for any given join.

For a true star join (that is, one in which none of the dimensions shares a join key with any other dimension), the IQ query optimizer allows a maximum of 24 dimension tables in a single clause. However, as the time required to process the query increases exponentially with the number of dimensions, performance degrades as you get close to this maximum.

See also

- *Entity and Referential Integrity* on page 355
- *Creating Foreign Keys* on page 187
- *Foreign Key Creation* on page 357

Restrictions on Modifying Join Index Tables

Once you have created a join index, you are restricted in the types of changes you can make to the join index and its underlying tables and indexes.

You cannot drop any table that participates in a join index. Likewise, you cannot use **ALTER TABLE** to add, drop, or modify a column that participates in a join index. In both cases, you must first drop the join index. Then you can either drop the table, or modify any columns that participate in the join index.

You can add columns to the tables that participate in a join index. However, there are restrictions on inserting data into these columns, as described in the next section.

You can drop indexes on columns not involved in the join relationship, and you can add, drop or modify nonjoined columns of tables in a join index. However, you cannot drop either the indexes on a join column or the join column itself. You need at least one index on a column involved in a predefined join relationship. It is highly desirable to have either an **HG** or **LF** index on all columns that are part of a join index.

Sybase IQ automatically applies the changes to the join index at the same time as it changes the base table. You do not need to synchronize the join index after any **ALTER TABLE** on nonjoined columns.

Other restrictions on **ALTER TABLE** for join indexes include the following:

- You cannot rename a column into or out of a **NATURAL** join condition.
- You cannot add a column that would participate in a previously specified **NATURAL** join.
- You cannot drop a **PRIMARY KEY/FOREIGN KEY** relationship if it matches a join condition that is in use in a join index.
- You cannot drop a **NOT NULL** constraint from a column that participates in a join condition.
- You cannot modify the data type of a column that participates in a join condition.

See also

- *Guidelines for Modifying Views* on page 190

Insertions or Deletions from Join Index Tables

You always insert or load into, or delete from, the underlying tables, not the join index itself.

When you first create the join index, Sybase IQ synchronizes the joined tables automatically, whether or not you have previously loaded data into the tables.

If you insert into or delete from a table that participates in an existing join index, you must synchronize the join index explicitly, unless you are updating the top table in the join hierarchy. If you insert rows and then delete them before the synchronization takes place, Sybase IQ optimizes synchronization to omit the insertions.

Privileges Needed to Manipulate Data in the Joined Tables

When you create a join index, you have the privileges necessary to perform operations on the join.

You must explicitly grant permissions on the underlying "join virtual table" to other users in your group, however, before they can manipulate tables in the join. These privileges must be granted on the join virtual table *in addition to* the appropriate privileges on the tables participating in the join.

Before granting privileges on the join virtual table, you must first determine its name, which is stored in a system table. If the name of the join index is `emp_location`, then the following query returns the name of the join virtual table:

```
select table_name from sys.systable
   where table_id in (select jvt_id from     sys.sysiqjoinindex
      where joinindex_name='emp_location')
```

Use the name of the join virtual table `jvt_name` returned by this query to grant permissions on the join virtual table:

```
grant all on jvt_name to user_names
```

After you grant the necessary privileges on the underlying join virtual table, other users in your group can perform operations on the tables in the join index without receiving permission errors.

Table Versioning Controls Access to Join Indexes

Any table is only available for write use to a single user at any given time.

For join indexes, this means that when one user is updating any table in a join index, no one else can update any of the tables in that index. All the joined tables remain unavailable until the first user's transaction is committed and you have synchronized the tables with the **SYNCHRONIZE** command.

Other users receive the following error while the join index tables are in use:

```
Cannot write to this table in current transaction.
Another user has write mode access.
```

Their current transactions cannot write to any of the join index tables; they must begin a new transaction to write to those tables.

See also

- *Intermediate Versioning* on page 17
- *Transactions and Versioning* on page 365

Size and Benefits of Join Indexes

Before creating a join index, estimate its size and potential benefit.

How to Estimate Join Index Size

Sybase IQ provides a stored procedure, **sp_iqestjoin**, to help you estimate the size of a join index.

You run this procedure for each pair of tables being joined. Each time you run the procedure, you must supply the following parameters:

- Name of the first table to be joined
- Number of rows in the first table
- Name of the second table to be joined
- Number of rows in the second table
- Relationship (default is one-to-many)
- IQ page size (default is 131072 bytes, or 128KB)

Many factors affect the size of a join index, especially the number of outer joins it includes. For this reason, the procedure offers you three types of results. If you know you will always join the tables with exact one-to-one matches, use the “Min Case index_size.” If you anticipate occasional one-to-many joins, use the “Avg Case index_size.” If you anticipate using numerous one-to-many joins, use the “Max Case index_size.”

These calculations should give you an idea of how much disk space you need for the join index. The results include the segment size in bytes, and the number of blocks. The procedure also tells you how long it will take to create the join index.

If you want to know the actual size of an existing join index, you use a different stored procedure, **sp_iqjoinindexsize**.

How to Determine Join Index Benefits

When considering creating a join index, compare the number of rows in the top table to the number of rows in the related table(s).

In general, a join index improves performance for many queries where the ratio of rows in the top table to rows in the related table is less than 10 to 1. Some users may find the join index advantageous with a ratio as high as 100 to 1, but others may find that the join index inhibits query performance if the ratio is as low as 10 to 1. If you are considering using a higher ratio, test to be sure it helps your queries.

Data Import and Export

Sybase IQ provides several methods of moving data into and out of your database.

You need to know when to use each method and how to address conversion issues for data inserted from other types of databases.

See also

• *Privileges Needed to Create a Join Index* on page 229

Import and Export Overview

Sybase IQ lets you import data from flat files or directly from database tables. You can also enter specified values directly into the database. Export of data to other formats is available from the Interactive SQL utility and the IQ data extraction facility.

An Sybase IQ table is a logical table; it does not contain data. All the information needed to resolve queries, including data, is contained in the Sybase IQ indexes. When you insert data into the columns in an IQ table, you are not actually adding data to the columns in the table, but rather to the column indexes. You build indexes by inserting data on a table-by-table basis.

Import and Export Method Selection

Sybase IQ offers you a choice of methods for adding, changing, or deleting data.

- For efficient bulk loading of tables from flat files, use the SQL statement **LOAD TABLE**.
- To insert specified values into a table row by row, use the SQL statement **INSERT** with the **VALUES** option.
- To insert rows selected from a table, use the SQL statement **INSERT** with a **SELECT** statement clause.
- To insert rows from a table residing in another database, use the SQL statement **INSERT** with a **LOCATION** statement clause.
- To remove specific rows from a table, use the **DELETE** statement. The **TRUNCATE** statement initializes a table to 0 rows.
- To change existing rows in a table, you can also use the **UPDATE** statement.

The IQ data extraction facility exports data in binary or ASCII format, which can be loaded into another database. Use this facility for high-volume data movement, or when you need an output file that can be used for loads.

From Interactive SQL, you can export data to another database in a variety of formats, or produce a text file as output. See the next section for a list of formats and how to select them. You can also redirect the output of any command.

Note: Sybase IQ supports BCP through the **LOAD TABLE FORMAT BCP** option. Sybase IQ also supports bulk loading of remote data using the **LOAD TABLE USING CLIENT FILE** option. See *Reference: Statements and Options*.

You can perform a BCP into a SQL Anywhere table, then transfer the contents to Sybase IQ; however, the transfer of rows is executed one row at a time. Sybase IQ does not support the Open Client Bulk-Library (blklib) routines, so BCP, which uses the Open Client Bulk-Library, does not work in bulk-load mode. (Bulk-Library/C provides routines that allow Client-Library and Server-Library applications to use Adaptive Server's bulk copy interface for high-speed data transfer.) The BCP format in both Sybase IQ and Adaptive Server supports a leading blank before a single digit that is part of a date.

See also

- *The Extract Options* on page 245

Input and Output Data Formats

The **LOAD TABLE** statement imports data from files row by row. Both ASCII and binary input files are supported, with either fixed-length fields or variable-length fields ended by a delimiter.

The **INSERT** statement moves data into a Sybase IQ table either from a specified set of values, or directly from tables.

Interactive SQL supports the following output file formats:

File Format	Description
ASCII	A text file, one row per line, with values separated by a delimiter. String values are optionally enclosed in apostrophes (single quotes). This is the same as the format used by **LOAD TABLE**
DBASEII	DBASE II format
DBASEIII	DBASE III format
EXCEL	Excel format
FIXED	Data records are in fixed format with the width of each column either the same as defined by the column's type or specified as a parameter
FOXPRO	FoxPro format
HTML	HTML format
LOTUS	Lotus worksheet format
SQL	Interactive SQL INPUT statement required to recreate the information in the table
XML	XML format encoded in UTF-8 and containing an embedded DTD.

The IQ data extraction facility exports data in binary or ASCII format.

See also

- *The Extract Options* on page 245

Specifying an Output Format for Interactive SQL

You can specify a default output format from Interactive SQL.

1. From the Interactive SQL menu bar, select Command > Options.
2. Choose an Output Format.
3. To make this the default output format, click Permanent.

Permissions for Modifying Data

You can only execute data modification statements if you have the proper permissions on the database tables you want to modify.

The database administrator and the owners of database objects use the GRANT and REVOKE statements to decide who has access to which data modification functions.

To insert data, you need INSERT permission for that table or view. To delete data, you need DELETE permission for that table or view. To update data, you need UPDATE permission. The DBA can insert into or delete from any table. The owner of a table has INSERT, DELETE, and UPDATE permission on it.

Permissions can be granted to and revoked from individual users, groups, or the public group.

See also

- *Managing User IDs and Permissions* on page 303
- *Security Overview* on page 4
- *Transactions and Versioning* on page 365

Schedule Database Updates

Multiple users can query a database table while one user inserts data into that table. Multiple users can update the database concurrently, as long as they are insert into or delete from different tables.

When you allow concurrent use of the database during updates, you pay a penalty in performance and disk use.

See also

- *Tune Bulk Loading of Data* on page 297

Methods for Exporting Data from a Database

To export IQ data from your database in this version of Sybase IQ, Sybase recommends that you use the methods documented in this guide.

You may also export data by using a front end tool, written by you or a third party, that effectively queries the IQ database and formats the data as desired.

To export tables (other than your system tables) from your catalog store, use the methods in this guide.

Output Redirection

Output redirection can be used to export query results.

You can redirect the output of any command to a file or device by putting the **>#** redirection symbol anywhere on the command. The redirection symbol must be followed by a file name. (In a command file, the file name is then followed by the semicolon used as statement terminator.) The file is placed relative to the directory where Interactive SQL was started.

In this example, output is redirected to the file `empfile`:

```
SELECT *
FROM Employees
># empfile
```

Do not enclose the file name in quotation marks.

Output redirection is most useful on the **SELECT** statement.

Use two **>** characters in a redirection symbol instead of one (for example, **>>#**), to append the output to the specified file instead of replacing the contents of the file. Headings are included in the output from the **SELECT** statement if the output starts at the beginning of the specified file and the output format supports headings.

Redirecting Output and Messages

The **>&** redirection symbol redirects all output including error messages and statistics for the command on which it appears. For example:

```
SELECT *
FROM Employees
>& empfile
```

Do not enclose the file name in quotation marks.

This example outputs the **SELECT** statement to the file `empfile`, followed by the output from the **SELECT** statement and some statistics pertaining to the command.

The *>&* redirection method is useful for getting a log of what happens during a **READ** command. The statistics and errors of each command are written following the command in the redirected output file.

NULL Value Output

Although the most common reason to extract data is for use in other software products, these products may sometimes have issues processing NULL values.

The **dbisql** option **NULLS** allows you to choose how NULL values are output. Alternatively, you can use the **IFNULL** function to output a specific value whenever there is a NULL value.

For information on setting **dbisql** options, see Chapter 2, “Database Options” in *Reference: Statements and Options*.

Data Extraction Facility

The data extraction facility is a group of database options that improve performance dramatically for queries with large result sets.

Like other database options, the data extraction options can be set either as temporary or permanent. Ordinarily these options are set as temporary. The extract options are set for a connection.

There are two advantages of using the extract options:

- A binary format is supported, which allows loading the output data into the same or a different IQ database.
- A **SELECT** statement with heavy output will run up to 4 times faster for ASCII output and up to 9 times faster for binary output.

The Extract Options

The extract options let you redirect the output of a **SELECT** statement from the standard interface to go directly to one or more disk files or named pipes.

Option Name	Allowed Values	Default value
Temp_Extract_Append	ON or OFF	OFF
Temp_Extract_Binary	ON or OFF	OFF
Temp_Extract_Column_Delimiter	string	','
Temp_Extract_Directory	string	''
Temp_Extract_Name1	string	''
Temp_Extract_Name2	string	''
Temp_Extract_Name3	string	''

Option Name	Allowed Values	Default value
Temp_Extract_Name4	string	''
Temp_Extract_Name5	string	''
Temp_Extract_Name6	string	''
Temp_Extract_Name7	string	''
Temp_Extract_Name8	string	''
Temp_Extract_Null_As_Empty	ON or OFF	OFF
Temp_Extract_Null_As_Zero	ON or OFF	OFF
Temp_Extract_Quote	string	''
Temp_Extract_Quotes	ON or OFF	OFF
Temp_Extract_Quotes_All	ON or OFF	OFF
Temp_Extract_Row_Delimiter	string	''
Temp_Extract_Size1	platform specific*	0
Temp_Extract_Size2	platform specific*	0
Temp_Extract_Size3	platform specific*	0
Temp_Extract_Size4	platform specific*	0
Temp_Extract_Size5	platform specific*	0
Temp_Extract_Size6	platform specific*	0
Temp_Extract_Size7	platform specific*	0
Temp_Extract_Size8	platform specific*	0
Temp_Extract_Swap	ON or OFF	OFF

*The default values for the Temp_Extract_Size*n* options are platform specific:

- AIX and HP-UX: 0 – 64GB
- Sun Solaris: 0 – 512GB
- Windows: 0 – 128GB
- Linux: 0 – 512GB

When large file systems, such as JFS2, support file size larger than the default value, set `TEMP_EXTRACT_SIZEn` to the value that the file system allows. For example, to support lTB set option:

```
SET OPTION TEMP_EXTRACT_SIZE1 = 1073741824 KB
```

Note: For all database options that accept integer values, Sybase IQ truncates any decimal *option-value* setting to an integer value. For example, the value 3.8 is truncated to 3.

The most important of these options is `TEMP_EXTRACT_NAME1`. If `TEMP_EXTRACT_NAME1` is set to its default setting (the empty string), extraction is disabled and no output is redirected. To enable extraction, set `Temp_Extract_Name1` to a possible path name. Extract starts extracting into a file with that name. Be sure to choose the path name to a file that is not otherwise in use. If the file does not already exist, the data extraction facility creates the file.

Both the directory or folder containing the named file and the named file must have write permission set for the user name used to start IQ (for example, *sybase*). In append mode, the data extraction facility adds extracted rows to the end of the file and does not overwrite the data that is already in the file. If the file does not exist, the data extraction facility creates the file.

Warning! If you choose the path name of an existing file and the `TEMP_EXTRACT_APPEND` option is set OFF (the default), the file contents are overwritten. This may be what you want if the file is for a weekly report, for example, but is not what you want if the file is one of your database files.

The options `TEMP_EXTRACT_NAME2` through `TEMP_EXTRACT_NAME8` are used in addition to `TEMP_EXTRACT_NAME1` to specify the names of multiple output files. These options must be used sequentially. For example, `TEMP_EXTRACT_NAME3` has no effect unless both the options `TEMP_EXTRACT_NAME1` and `TEMP_EXTRACT_NAME2` are already set.

The options `TEMP_EXTRACT_SIZE1` through `TEMP_EXTRACT_SIZE8` are used to specify the maximum size of the corresponding output files. `TEMP_EXTRACT_SIZE1` specifies the maximum size of the output file specified by `TEMP_EXTRACT_NAME1`, `TEMP_EXTRACT_SIZE2` specifies the maximum size of the output file specified by `TEMP_EXTRACT_NAME2`, and so on.

Note that the default for the data extraction size options is 0. IQ converts this default to the following values:

Device type	Size
disk file	AIX and HP-UX: 0 – 64GB Sun Solaris & Linux: 0 – 512GB Windows: 0 – 128GB
tape*	524288KB (0.5GB)
other	unlimited

*Tape devices currently are not supported.

TEMP_EXTRACT_APPEND is not compatible with the TEMP_EXTRACT_SIZEn options. If you try to restrict the size of the extract append output file, Sybase IQ reports an error.

If you are extracting to a single disk file or a single named pipe, leave the options TEMP_EXTRACT_NAME2 through TEMP_EXTRACT_NAME8 and TEMP_EXTRACT_SIZE1 through TEMP_EXTRACT_SIZE8 at their default values.

Note: If the **SELECT** returns no rows and there is no output to redirect, an empty file of zero length is created. If multiple extract files are specified and there is not enough data to fill all of the files, all of the files are still created.

Controlling Access

The TEMP_EXTRACT_DIRECTORY option controls whether a user is allowed to use the data extraction facility. It also controls the directory into which temp extract files are placed and overrides a directory path specified in the TEMP_EXTRACT_NAMEn options.

If the TEMP_EXTRACT_DIRECTORY option is set to the string FORBIDDEN (case insensitive) for a user, then that user is not allowed to perform data extracts. An attempt by this user to use the data extraction facility results in an error: “You do not have permission to perform Extracts”.

If TEMP_EXTRACT_DIRECTORY is set to FORBIDDEN for the PUBLIC group, then no one can run data extraction.

If TEMP_EXTRACT_DIRECTORY is set to a valid directory path, temp extract files are placed in that directory, overriding a path specified in the TEMP_EXTRACT_NAMEn options.

If TEMP_EXTRACT_DIRECTORY is set to an invalid directory path, an error occurs: “Files does not exist File: <invalid path>”

If TEMP_EXTRACT_DIRECTORY is blank, then temp extract files are placed in directories according to their specification in TEMP_EXTRACT_NAMEn. If no path is specified as part of TEMP_EXTRACT_NAMEn, the extract files are by default placed in the server startup directory.

The TEMP_EXTRACT_DIRECTORY option provides increased security and helps control disk management by restricting the creation of large data extraction files to the directories for which a user has write access. DBA authority is required to set this option.

Types of Extraction

There are three types of data extraction:

- Binary
- Binary/swap
- ASCII

A binary extraction produces a file that can be loaded via a **LOAD TABLE** statement with an overall "binary" format and with a per column "binary with null byte" format.

The binary/swap extraction is the same as the binary extraction, except it is designed to be loaded on another machine with opposite endianness.

The ASCII extraction produces a text file.

The two options Temp_Extract_Binary and Temp_Extract_Swap determine which of the three types of extraction is done:

Type	Temp_Extract_Binary	Temp_Extract_Swap
binary	ON	OFF
binary/swap	ON	ON
ASCII	OFF	OFF

The default extraction type is ASCII.

Note that if the data is unloaded using the extraction facility with the `TEMP_EXTRACT_BINARY` option set ON, then you must use the **LOAD TABLE** statement **BINARY WITH NULL BYTE** parameter for each column when you load the binary data.

Column and Row Delimiters

In the case of an ASCII extraction, the default is to separate column values with commas, and end the row with a newline on UNIX platforms and with a carriage return/newline pair on Windows platforms. The strings are unquoted. If these defaults are not suitable, use the following options to change the delimiters:

- Temp_Extract_Column_Delimiter
- Temp_Extract_Row_Delimiter
- Temp_Extract_Quote
- Temp_Extract_Quotes
- Temp_Extract_Quotes_All

The delimiter must occupy from 1 to a maximum of 4 bytes and must be valid in the collation order you are using, if you are using a multibyte collation order. Be sure to choose delimiters that do not occur in any of the data output strings themselves.

Note that the default for the Temp_Extract_Row_Delimiter option is the empty string. IQ converts the empty string default for this option to the newline on UNIX platforms and to the carriage return/newline pair on Windows platforms.

The option Temp_Extract_Column_Delimiter controls the delimiter between columns. If this option is set to the empty string '' for ASCII extractions, then the extracted data is written in fixed-width ASCII with no column delimiter. Numeric and binary data types are right-justified on a field of *n* blanks, where *n* is the maximum number of bytes needed for any value of that type. Character data types are left-justified on a field of *n* blanks.

Note: The minimum column width in a fixed-width ASCII extraction is four bytes to allow the string “NULL” for a NULL value. For example, if the extracted column is `CHAR(2)` and

Temp_Extract_Column_Delimiter is set to the empty string '', there are two spaces after the extracted data.

During ASCII extraction, the following options control the use of quotes:

Option	ASCII extraction action
Temp_Extract_Quotes	string fields enclosed in quotes
Temp_Extract_Quotes_All	all fields enclosed in quotes
Temp_Extract_Quote	specifies string to be used as the quote

The quote string specified in the Temp_Extract_Quote option has the same restrictions as delimiters. The default for this option is the empty string, which IQ converts to the single quote mark.

Representation of Null Values

The Temp_Extract_Null_As_Zero and Temp_Extract_Null_As_Empty options controls the representation of null values for ASCII extractions. When the Temp_Extract_Null_As_Zero option is set to ON, a null value is represented as follows:

- '0' for arithmetic type
- '' (the empty string) for the CHAR and VARCHAR character types
- '' (the empty string) for dates
- '' (the empty string) for times
- '' (the empty string) for timestamps

When the Temp_Extract_Null_As_Empty option is set to ON, a null value is represented as '' (the empty string) for all data types.

Note that the quotes shown above are not present in the extract output file. When the Temp_Extract_Null_As_Zero and Temp_Extract_Null_As_Empty options are set to OFF, the string 'NULL' is used in all cases to represent a NULL value. OFF is the default value.

If Temp_Extract_Null_As_Zero is set to ON, the number of characters that an ASCII extract writes to a file for a `CHAR` or `VARCHAR` column equals the number of characters in the column, even if that number is less than four. In previous releases, Sybase IQ always returned at least four characters to accommodate the word NULL.

Message Logging

When the Query_Plan option is ON, a timestamped list of the extracted columns appears in the IQ message log.

See also

- *Import and Export Method Selection* on page 241
- *Input and Output Data Formats* on page 242

Enabling Data Extraction Options

Use the data extraction options with care.

Warning! If you set the extract options, then execute a **SELECT** statement, and then execute a second **SELECT** statement without changing the extract file name, the output of the second **SELECT** overwrites the output of the first **SELECT**. Each time you execute a **SELECT** statement, whether it is one second later or a week later, extract starts over again, unless the Temp_Extract_Append option is set ON.

Also be aware that the extract options are set for the connection. If you set the extract options and another user connects to the database using the same connection, the extract facility is also enabled for that user. Your extraction output can be overwritten by another user on the same connection.

Similarly, if another user logs in using the same user ID, the output of queries run by this user is directed to the extract file until the option is disabled. If you are using extract, be sure to run your request from a unique user ID.

---

1. In a separate location, save any old output extract you need to retain.
2. Remove any previously used extract files.
3. Set the extraction options you require, making sure to set Temp_Extract_Name1 to the file path that is to receive the output.
4. Issue a **SELECT** statement to extract the data you require.
5. Reset Temp_Extract_Name1 to the empty string, or disconnect if set temporarily, when you are finished making extractions.

Data Extraction Option Examples

These examples describe various data extraction scenarios.

Example: Extracting to a Single Disk File

The following statements extract to a single disk file `daily_report.txt`:

```
SET TEMPORARY OPTION Temp_Extract_Name1 = 'daily_report.txt';

SET TEMPORARY OPTION Temp_Extract_Name2 = '';

SELECT ....;

SET TEMPORARY OPTION Temp_Extract_Name1 = '';
```

Note that Temp_Extract_Name2 is set to the empty string before the **SELECT** statement is executed, to restrict output to a single file.

Also note that Temp_Extract_Name1 is set to the empty string after the **SELECT** statement to disable extraction. If extraction is not disabled, then the next **SELECT** statement executed overwrites the `daily_report.txt` file.

Example: Extracting in Append Mode

In this example, the disk output file `hourly_report.txt` is already created and has write permission set for the user *sybase*. The following statements extract to `hourly_report.txt`, appending the output from each **SELECT** statement to the end of the file:

```
SET TEMPORARY OPTION Temp_Extract_Append = ON;
SET TEMPORARY OPTION Temp_Extract_Name1 = 'hourly_report.txt';
SET TEMPORARY OPTION Temp_Extract_Name2 = '';
SELECT ....;
SELECT ....;
SELECT ....;
SET TEMPORARY OPTION Temp_Extract_Name1 = '';
```

All of the output from the three **SELECT** statements is written to the file `hourly_report.txt`. Temp_Extract_Name1 is set to the empty string after the last **SELECT** statement to disable extraction. If extraction is not disabled, then the output from the next **SELECT** statement executed is also added to the end of the file `hourly_report.txt`.

Example: Extracting to Multiple Disk Files

The following statements extract to disk files `file1.out`, `file2.out`, and `file3.out`.

First set the file name options:

```
SET TEMPORARY OPTION Temp_Extract_Name1 = 'file1.out';
SET TEMPORARY OPTION Temp_Extract_Name2 = 'file2.out';
SET TEMPORARY OPTION Temp_Extract_Name3 = 'file3.out';
SET TEMPORARY OPTION Temp_Extract_Name4 = '';
```

Now limit the size of the files, for example to 1MB each, by setting the corresponding extract size options:

```
SET TEMPORARY OPTION Temp_Extract_Size1 = '1024';
SET TEMPORARY OPTION Temp_Extract_Size2 = '1024';
SET TEMPORARY OPTION Temp_Extract_Size3 = '1024';
```

The size options are in KB (1024 bytes).

With these settings, the extraction output is first written to `file1.out`. When the next row to be written to `file1.out` would cause the file size to exceed 1MB, the output is redirected to `file2.out`. When `file2.out` is full (writing another row to `file2.out` would cause the file size to exceed 1MB), the output is redirected to `file3.out`. An error is reported, if the size of `file3.out` exceeds 1MB before IQ extracts all rows.

Extraction Limitations

These restrictions and limitations affect the data extraction facility.

- Extract works only with data stored in the IQ store.

- Extract does not work on system tables or cross database joins.
- Extract does not work with queries that use user-defined functions or system functions, except for the system functions **suser_id()** and **suser_name()**.
- A binary **LOAD TABLE** always trims blanks from VARCHAR data. If you have VARCHAR data with trailing blanks, they are not preserved on insert by a binary load.
- Trailing zeros are padded onto VARBINARY data during the extract. For example, a field declared as varbinary(6), which contains the data 0x1234, is padded with zeros during extraction and is loaded after extraction as 0x123400.
- Binary format will change in a future release.
- If you need to reproduce floating point data exactly, use the binary option.
- Tape devices are not supported at this time.
- If you run **dbisql** (Interactive SQL) with the **-q** (quiet mode) option and the data extraction commands are in a command file, you must first set and make permanent the **dbisql** option **Show multiple result sets**. If this option is not set, the output file is not created.

  To set the **Show multiple result sets** option, click **Tools > Options** in the **dbisql** window, then check the box **Show multiple result sets** and click **Make permanent**.

Also note that when Temp_Extract_Name1 is set, you cannot perform these operations:

- **LOAD**, **DELETE**, **INSERT** or **INSERT...LOCATION** to a table that is the top table in a join
- **SYNCHRONIZE JOIN INDEX** (issued explicitly or executed as part of **CREATE JOIN INDEX**)
- **INSERT...SELECT**

Events do not support execution of statements that return result sets. The server log returns an error similar to the following:

```
Handler for event 'test_ev' caused SQLSTATE '09W03'
Result set not permitted in 'test_ev'
```

In order to execute a query through an event, create an event that calls a stored procedure and insert the stored procedure results into a temporary table. If extract is used, then the temporary table is always empty and requires little overhead.

For example:

```
CREATE PROCEDURE proc1()
BEGIN
    SET TEMPORARY OPTION temp_extract_name1 = 'testproc.out';
    SELECT * FROM iq_table;
END;

CREATE EVENT "test_ev" ENABLE HANDLER
BEGIN
    SELECT * INTO #tmp FROM proc1();
END;

TRIGGER EVENT test_ev;
```

Bulk Loads with the LOAD TABLE Statement

The **LOAD TABLE** statement efficiently imports data from a text or binary file into an existing database table.

The statement loads data into column indexes created automatically or defined by users.

The permissions needed to execute a **LOAD TABLE** statement are set on the server command line, using the **-gl** option. Sybase recommends the **-gl all** setting, which allows any user to load or unload a table. This is the default setting set by **start_iq**. If **-gl all** is set, you must be the owner of the table, have DBA authority, or have ALTER permission, in order to use the **LOAD TABLE** statement. You also need INSERT permission on the table.

Using Command Files to Load Data

To load large amounts of data, most users create command files.

Transaction Processing and LOAD TABLE

When you issue the **LOAD TABLE** statement for an IQ table, a savepoint occurs automatically before the data is loaded.

If the load completes successfully, Sybase IQ releases the savepoint. If the load fails, the transaction rolls back to the savepoint. This approach gives you flexibility in committing transactions. For example, if you issue two **LOAD TABLE** commands, you can ensure that either both commands commit or neither commits.

When you issue **LOAD TABLE** for a catalog store table, there is no automatic savepoint. If the load succeeds, it commits automatically. If the load fails, it rolls back. You cannot roll back a successful load of a catalog store table.

Load from a Flat File: UNIX Example

The following statement loads the data from the file `dept.txt` into all columns of the `department` table.

This example assumes that no explicit data conversion is needed, and that the width of input columns matches the width of columns in the `Departments` table. The flat file `dept.txt` must exist at the specified location.

```
LOAD TABLE Departments
( DepartmentID, DepartmentName, DepartmentHeadID )
FROM '/d1/MILL1/dept.txt'
```

File Specification Requirements for Loads

In the **FROM** clause, use *filename-string* to specify files, and separate multiple strings by commas.

The files are read one at a time, and processed in a left-to-right order as specified in the **FROM** clause. Any **SKIP** or **LIMIT** value only applies in the beginning of the load, not for each file.

If a load cannot complete, for example due to insufficient memory, the entire load transaction rolls back.

The *filename-string* is passed to the server as a string. The string is therefore subject to the same formatting requirements as other SQL strings. In particular:

- If a backslash (\) precedes the characters `n`, `x`, or `\` it is considered an escape character. For this reason, to indicate directory paths in Windows systems, you must represent the backslash character by two backslashes if the next character is any of those listed. (It is always safe to double the backslashes.)
  To load data from the file `c:\newinput.dat` into the `employee` table:
  ```
  LOAD TABLE employees
  FROM 'c:\\newinput.dat' ...
  ```
- For server-side loading (**LOAD TABLE ... USING FILE**), the path name is relative to the database server, not to the client application. If you are running the statement on a database server on some other computer, the directory name refers to directories on the server machine, not on the client machine. The input file for the load must be on the server machine.
- For client-side data loading (**LOAD TABLE ... USING CLIENT FILE**), the path name must be relative to the client application. The directory name refers to directories on the client machine.

Loads that Specify Named Pipes

When you load from a named pipe (or FIFO) on Windows, the program writing to the pipe must close the pipe in a special way.

The pipe writer must call `FlushFileBuffers( )` and then `DisconnectNamedPipe( )`. (If the program does not, Sybase IQ reports an exception from `hos_io::Read( )`.) This issues a `PIPE_NOT_CONNECTED` error, which notifies Sybase IQ that the pipe was shut down in an orderly manner rather than an uncontrolled disconnect. See Microsoft documentation for details on these calls.

See also

- *Inserting Data Directly from Adaptive Server Enterprise* on page 271
- *Load Time Environment Adjustments* on page 298
- *Tools for Investigating Lock Contention* on page 380

Loads that Specify Input Data Format

You can specify a wide range of load options that tell Sybase IQ how to interpret and process the input file and what to do when errors occur.

You can specify load options in any order. For details of all options, see the **LOAD TABLE** statement in *Reference: Statements and Options*.

Example: Load that Displays Quotation Marks

Consider a table defined as:

```
CREATE TABLE t1 (c1 INT, c2 VARCHAR(20), c3 VARCHAR(20))
```

with the following input data:

```
1, apple , fruit1      ,
2, "banana" , "fruit2",
3, " pear ", " fruit3 ",
```

The result of loading this data is displayed by running the following query:

```
SELECT c1, c2, c3, LENGTH(c2), LENGTH(c3) FROM t1
```

Given the values of the **QUOTES** and **STRIP** options used during the **LOAD TABLE** command, the following table displays the result of the query above, with each result enclosed by ‘<’ and ‘>’:

LOAD TABLE options		Results of SELECT c1, c2, c3, LENGTH(c2), LENGTH(c3) FROM t1				
QUOTES	**STRIP**	**c1**	**c2**	**c3**	**length(c2)**	**length(c3)**
ON	RTRIM	<1>	<apple>	<fruit1>	<5>	<6>
		<2>	<banana>	<fruit2>	<6>	<6>
		<3>	< pear >	< fruit3 >	<6>	<8>
ON	OFF	<1>	<apple >	<fruit1 >	<6>	<7>
		<2>	<banana>	<fruit2>	<6>	<6>
		<3>	< pear >	< fruit3 >	<6>	<8>
OFF	RTRIM	<1>	< apple>	< fruit1>	<6>	<7>
		<2>	< “banana”>	< “fruit2”>	<9>	<9>
		<3>	< “ pear ”>	< “ fruit3 ”>	<9>	<11>
OFF	OFF	<1>	< apple >	< fruit1 >	<7>	<8>
		<2>	< “banana” >	< “fruit2”>	<10>	<9>
		<3>	< “ pear ”>	< “ fruit3 ”>	<9>	<11>

Notes on the results:

- With **QUOTES ON** and **STRIP RTRIM**, both leading space and trailing space for the non-enclosed field `c2` row 1 are trimmed.
- With **QUOTES ON** and **STRIP OFF**, only the leading space for the non-enclosed field `c2` row 1 is trimmed.
- With **QUOTES OFF** and **STRIP RTRIM**, only the trailing space for the non-enclosed field `c2` row 1 is trimmed.
- With **QUOTES OFF** and **STRIP OFF**, neither leading space nor trailing space for the non-enclosed field `c2` row 1 is trimmed.
- With **QUOTES ON** and **STRIP RTRIM**, both leading space and trailing space within quotes for the enclosed fields `c2` and `c3` row 3 are NOT trimmed.

Example: Load that Skips Specified Fields.

The following is a Windows example:

```
LOAD TABLE nn
    (l_orderkey,
    l_quantity ASCII(PREFIX 2),
    FILLER(2),
FROM 'C:\\iq\archive\\mill.txt'
BYTE ORDER LOW
```

Example: Load that Limits the Number of Rows Inserted

In the following Windows example, no rows are skipped and up to 1,000,000 rows are inserted.

```
LOAD TABLE lineitem
    (l_shipmode ASCII(15),
    l_quantity ASCII(8),
    FILLER(30))
FROM 'C:\\iq\archive\\mill.txt'
PREVIEW ON
LIMIT 1000000
```

Example: Load that Includes Tabs and New Lines

The following Windows example sets the column delimiter for the l_orderkey column to tab, and the row delimiter to newline (\x0a) followed by carriage return (\x0d):

```
LOAD TABLE mm
    (l_orderkey '\x09',
    l_quantity ASCII(4),
    FILLER(6),
    l_shipdate DATE('YYYY/MM/DD'))
FROM 'C:\\iq\\archive\\mill.txt'
ROW DELIMITED BY '\x0a\x0d'
```

Example: Load That Skips Rows

In this UNIX example, Sybase IQ reads 9,000 rows from the input file, skips the first 5,000, and loads the next 4,000. If there are only 8,000 rows in the input file, then only 3,000 rows are loaded.

```
LOAD TABLE lineitem(
    l_shipmode ASCII(15),
    l_quantity ASCII(8),
FILLER(30))
FROM '/d1/MILL1/tt.t'
LIMIT 4000
SKIP 5000
PREVIEW ON
```

LOAD TABLE Adds Rows

The **LOAD TABLE** statement appends the contents of the file to the existing rows of the table; it does not replace the existing rows in the table, unless you specify the **START ROW ID** load option.

To empty an existing table, use the **TRUNCATE TABLE** statement to remove all the rows.

See also

- *Load Conversion Options* on page 277

Direct Loading of Data from Clients

Sybase IQ supports bulk loading of remote data via the **LOAD TABLE USING CLIENT FILE** statement. **LOAD TABLE USING FILE** loads data on the local server and replaces the deprecated utility **iq_bcp**.

Both the server and client must be Sybase IQ version 15.0 or later. A combination of 15.0 or later server and a 12.7 client returns a File I/O Error.

For syntax, see the **USING** clause in the **LOAD TABLE** statement in *Reference: Statements and Options*.

Considerations for Partitioned Table Loads

There are a few special considerations for loading partitioned tables:

- When you modify a partitioned table, you can achieve the best performance when the partitioning column is the first column in the column list of the command.

  List partitioning columns before large object (LOB) columns in the **SELECT** statement clause of an **INSERT...LOCATION** statement and load data from a primary file. If possible, use a preload process to rearrange data in the primary file.
- The **START ROW ID** clause of the **LOAD TABLE** and the **INSERT** statements is not allowed on a partitioned table. The following error is reported and a rollback is performed on the load operation:

  ```
  "Option START ROW ID not allowed on a partitioned table."
  ```

  ```
  (SQLCODE -1009416L, SQLSTATE QCB14, Sybase error code 21054)
  ```
- The `APPEND_LOAD` database option behaves differently for partitioned and unpartitioned tables. Row ID ranges are assigned to each partition.

For partitioned tables, when `APPEND_LOAD` is ON, new rows are appended at the end of the appropriate partition. When `APPEND_LOAD` is OFF, the load reuses the first available row IDs and space from deleted rows of the appropriate partition.

For non-partitioned tables, when `APPEND_LOAD` is ON, new rows are added after the maximum row ID that is at the end of the table rows. When `APPEND_LOAD` is OFF, the load reuses the deleted row IDs. You can also use the **LOAD** or **INSERT START ROW ID** clause to specify the row at which to start inserting.

- Attempting to update the contents of a partitioning column results in the following error:

```
"Updating partition key column on a partitioned table is not
allowed."
```

```
(SQLCODE -1009417L, SQLSTATE QCB15, Sybase error code 21055)
```

Table partitioning is part of the separately licensed Sybase IQ VLDB Management option.

See also

- *Restrictions on DDL Operations on Partitions* on page 144

Load and Insert Messages

You can use a database option and a server startup switch to control insert and load messages.

Messages may display during insert and load operations. The `NOTIFY_MODULUS` database option adjusts the default frequency of notification messages during loads, or omits these message. The `NOTIFY` option in the **LOAD** command overrides the `NOTIFY_MODULUS` setting. See *Reference: Statements and Options*.

The **IQMsgMaxSize** server property and the **-iqmsgsz** server startup switch control message log wrapping and the size of the message log file.

See also

- *Columns in the Join Index* on page 225
- *Interpreting Notification Messages* on page 515
- *Message Log Wrapping* on page 11
- *Version Information in Message Logs* on page 9

Integrity Constraint Violation Messages

LOAD TABLE allows you to control load behavior when integrity constraints are violated and to selectively log information about the violations.

Using the **MESSAGE LOG ... ROW LOG** option with the **ONLY LOG** clause, you can direct the load to log information about specific types of integrity constraint violations both per violation in a message log file and per row in a row log file. If the **ONLY LOG** clause is not specified, only the timestamps indicating the start and completion of the load are logged in these files.

Note that the message log and row files for integrity constraint violations are distinct from the IQ message log file (`.iqmsg`).

You can specify whether to ignore UNIQUE, NULL, DATA VALUE, and/or FOREIGN KEY constraint violations that occur during a load and the maximum number of violations to ignore before initiating a rollback. You can also direct the load to log information about specific types of integrity constraint violations both per violation in a message log and per row in a row log.

MESSAGE LOG Contents and Format

The **MESSAGE LOG** file contains row and column information for each integrity constraint violation logged.

For a given load, there are three types of messages logged: a timestamped header, row information, and a timestamped trailer. The header appears once per load. The tailer appears once if the statement executes successfully. The row information appears once for each integrity constraint violation logged.

The format of the header message is as follows:

```
<datetime load started> Load Table <table-name>: Integrity
Constraint Violations
```

For example:

```
2009-05-24 23:04:31 Load Table Customers: Integrity Constraint
Violations
```

The row information message consists of three parts:

- The row number within the table where this row would have been loaded, if an integrity constraint violation had not occurred.
- The type of integrity constraint violation detected.
- The column number(s) specified by the schema.

For example,

```
1267 DATA VALUE 4
3216 UNIQUE 1
3216 NULL 3
3216 NULL 6
9677 NULL 1
```

The format of the trailer message is as follows:

```
<datetime load completed> Load Table <table-name> Completed
```

For example:

```
2009-05-24 23:05:43 LOAD TABLE Customers: Completed
```

Note: The number of rows (errors reported) in the **MESSAGE LOG** file can exceed the **IGNORE CONSTRAINT** option limit, because the load is performed by multiple threads running in parallel. More than one thread may report that the number of constraint violations has exceeded the specified limit.

ROW LOG Contents and Format

The **ROW LOG** file contains rowid and data values for each row on which logged integrity constraint violation(s) occurred.

The row data appears exactly once for a given row, regardless of the number of integrity constraint violations that occurred on that row. For a given load, there are three types of messages logged: a timestamped header, row data, and a timestamped trailer. The header appears once per load. The tailer appears once if the statement executes successfully.

The format of the header message is as follows:

```
<datetime load started> Load Table <table-name>: Integrity
Constraint Violations
<formatting information>
```

where `<formatting information>` is the date, time, and datetime formats used in formatting the row data. For example:

```
2009-05-24 23:04:31 Load Table Customers: Integrity Constraint
Violations
Date Format: yyyy/mm/dd
Time Format: hh:mm:ss
Datetime format: yyyy/mm/dd hh:mm:ss
```

The row data message consists of two parts:

- The row number within the table where this row would have been loaded, if an integrity constraint violation had not occurred.
- The data values in the row, separated by either a comma or the user-specified **LOG DELIMITED BY** separator.

For example,

```
3216 #Jones John#NULL#NULL#S#1945/01/12#NULL#
```

The format of the data values in the row data message is determined by the following rules:

- When the data type is `VARBINARY` or `BINARY`, the data is represented by ASCII hexadecimal characters.
- `DATE` values are represented in the format specified by the DATE_FORMAT database option. The default format is YYYY-MM-DD.
- `DATETIME` and `TIMESTAMP` values are represented in the format specified by the TIMESTAMP_FORMAT database option. The default is YYYY-MM-DD HH:NN:SS.SSS.
- `TIME` values are represented in the format specified by the TIME_FORMAT database option. The default is HH:NN:SS.SSS.
- NULL values are represented by the token NULL.

Note: Filler fields do *not* appear in the row data message.

The format of the trailer message is as follows:

```
<datetime load completed> Load Table <table-name>: Completed
```

For example:

```
2009-05-24 23:05:43 Load Table Customers: Completed
```

Note: The number of distinct errors in the **MESSAGE LOG** file may not exactly match the number of rows in the **ROW LOG** file. The difference in the number of rows is due to the parallel processing of the load performed by multiple threads. More than one thread may report that the number of constraint violations has exceeded the specified limit.

MESSAGE LOG and ROW LOG Example

This example illustrates the contents and format of the **MESSAGE LOG** and **ROW LOG** files.

The following **CREATE TABLE** statement creates the table that is loaded using a **LOAD TABLE** statement:

```
CREATE TABLE Customers(name VARCHAR(80) NOT NULL,
age TINYINT NULL,
sex CHAR(1) NOT NULL,
marital_status CHAR(1) NULL,
birthdate DATE NOT NULL,
credit_card VARCHAR(20)NOT NULL)
```

The following **LOAD TABLE** statement loads the data into the `Customers` table:

```
LOAD TABLE Customers ...
IGNORE CONSTRAINT UNIQUE 200
MESSAGE LOG 'msg.log' ROW LOG 'row.log'
ONLY LOG UNIQUE, NULL, DATA VALUE
LOG DELIMITED BY '#'
```

The following raw data is loaded from a disk file using the **LOAD TABLE** statement above:

```
Jones John, 19, M, S, 06/19/83, CC
Cleven Bill, 56, M, OSIDJFJ, 02/23/43, CC
Jones John, 339, M, NULL, 01/12/45, NULL
NULL, 55, F, M, 10/02/37, ST
```

After the **LOAD TABLE** completes, the **MESSAGE LOG** file `msg.log` contains the following information:

```
2009-05-24 23:04:31 LOAD TABLE Customers: Integrity Constraint
Violations
1267 DATA VALUE 4
3216 UNIQUE 1
3216 NULL 6
9677 NULL 1
2009-05-24 23:05:43 LOAD TABLE Customers Completed
```

After the **LOAD TABLE** completes, the **ROW LOG** file `row.log` contains the following information:

```
2009-05-24 23:04:31 LOAD TABLE Customers Integrity Constraint
Violations
Date Format: yyyy/mm/dd
Time Format: hh:mm:ss
```

```
Datetime format: yyyy/mm/dd hh:mm:ss

1137 #Jones John#19#M#S#1983/06/19#CC#
1267 #Cleven Bill#56#M#OSIDJFJ#1943/02/23#CC#
3216 #Jones John#NULL#NULL#S#1945/01/12#NULL#
9677 #NULL#55#F#M#1937/10/02#ST#

2009-05-24 23:05:43 LOAD TABLE Customers Completed
```

Binary Load Formats

To speed data loading into Sybase IQ, create data files in Sybase IQ binary format, then load this data into Sybase IQ using the **FORMAT BINARY** and **BINARY** column specification clauses of **LOAD TABLE**.

You can find instructions for using **LOAD TABLE** in *Reference: Statements and Options*.

Create data files with these binary formats to load into columns with the corresponding data types. In most cases, Sybase IQ uses the platform-specific binary format. The following data types are exceptions that use binary formats that are specific to Sybase IQ:

- `DATE`
- `TIME`
- `DATETIME`
- `NUMERIC`

IQ Binary Load Format and Load Efficiency

The Sybase IQ binary load format is a fixed-width format.

In general, fixed-width loads complete faster than variable width loads. When the load logic recognizes the length of a column and row, the data is processed more efficiently. Using delimiters to separate columns and rows that vary in width forces the load to spend time scanning the input data looking for them.

The IQ Binary Load Format is a fixed-width load. The load can determine the width of each column and length of each row from information in the table definition.

Note: Binary load format is endian-sensitive. This is because binary load format utilizes native binary data types to represent data.

Operating System Native Data Types

Data for the following data types is stored in native operating system binary format and can be written to data files directly in that format. Sybase IQ reads the respective number of bytes directly into the associated data types without conversion.

- `BIT` (1 byte)
- `TINYINT` (1 byte)

- SMALLINT (2 bytes)
- INT/UNSIGNED INT (4 bytes)
- BIGINT/UNSIGNED BIGINT (8 bytes)
- FLOAT (4 bytes)
- DOUBLE (8 bytes)
- CHAR/VARCHAR (character data)
- BINARY/VARBINARY (binary data)

By default, VARCHAR and VARBINARY columns are read in as many bytes as specified by **LOAD TABLE** *column-spec*.

DATE

DATE column data is stored in Sybase IQ as four bytes (a 32-bit unsigned integer) representing the number of days since 0000-01-01.

To convert a calendar date to the Sybase IQ binary format, for a given year, month, and day, use:

```
year = current_year - 1;
days_in_year_0000 = 366;
binaryDateValue = (year * 365)
+ (year / 4)
- (year / 100)
+ (year / 400)
+ days_in_year_0000
+ day_of_current_year
-1;
```

For the *day_of_current_year* value in the formula above, consider the following example: February 12 is day 43.

TIME

TIME data is stored as a 64-bit unsigned quantity that represents a number in microseconds (in other words, 1.0e-6 seconds).

Compute the microsecond quantity for a given hour, minute, second, and microsecond (*usec*):

```
binaryTimeValue = (hour * 3600 + minute * 60 + second + microsecond )
* 1000000
```

TIMESTAMP

TIMESTAMP data is stored as a 64-bit unsigned integer and represents a quantity in microseconds. You can compute a binary TIMESTAMP value for a given year, month, day, hour, minute, second, and microsecond as follows:

Compute *binaryDateValue* for the date as shown above.

Compute *binaryTimeValue* for the time as shown above.

```
binaryDateTimeValue = binaryDateValue *
    86400000000 + binaryTimeValue
```

NUMERIC and DECIMAL

Formats for `NUMERIC` and `DECIMAL` data types vary as a function of precision.

The value must be right-padded with zeros to the full scale of the value. The value must also be fully left-padded with zeros, but, with binary programming, padding happens automatically. Once the values are padded, the decimal point is removed. For example, the value 12.34 looks like:

- `NUMERIC`(4,2): 1234
- `NUMERIC`(6,4): 123400
- `NUMERIC`(8,4): 00123400
- `NUMERIC`(12,6): 000012340000
- `NUMERIC`(16,8): 0000001234000000

After the value is padded and the decimal point is removed, these rules apply:

- If precision <= 4, binary format is identical to native operating system binary format for 2-byte integer quantity.
- If precision is between 5 and 9, binary format is identical to native operating system binary format for a 4-byte integer quantity.
- If precision is between 10 and 18, binary format is identical to native operating system binary format for an 8-byte integer quantity.
- If precision >= 19, there is a special format that uses this C++ struct definition:

  ```
  struct {
  unsigned char sign; // sign 1 for +, 0 for -
  unsigned char ndig; // # digits
  unsigned char exp; // exponent
  unsigned char erracc;  // should be 0
  unsigned short digits[80];
  };
  ```

  Exponent is excess-80 form, unless the value is zero. A “zero” value is represented as:

  ```
  sign = 1
  ndig = 0
  erracc = 0
  exp = 0
  ```

  The maximum exponent value is 159. The maximum number of supported digits is 288. “digits[0]” contains the least-significant digits. Digits are stored in a packed representation with two digits per “unsigned short” (2-byte) quantity. For a given “digit:”

  - `lower order digit = digit[i] & 0x00FF`
  - `high order digit = digit[i] & 0xFF00`

  For example, consider the value 100 represented as `NUMERIC`(20). The binary layout of this value is:

```
0x0101 0x5000 0x0064 0x0000 0x0000 ......

Sign = 0x01
Number digits = 0x01
Exponent = 0x50
Erracc = 0x00
Digits = 0x0064
```

As another example, consider the value 32769:

```
0x0102 0x5000 0x0ad1 0x0003 0x0000 0x0000 ....

Sign = 0x01
Number digits = 0x02
Exponent = 0x50
Erracc = 0x00
Digits = 0x0ad1 0x0003
```

If you translate the digits into base 10, you have:

```
0x0ad1 = 2769  0x0003 = 3
```

NULL Value Loads

The most expedient way to insert NULL values is to use the NULL byte in the input file and specify **WITH NULL BYTE** in the column specification of the **LOAD TABLE** statement.

This is done by terminating each data field in the input file with “x00” or “x01”. Terminating a data field in the input file with “x01” instructs the load to insert NULL into the column. For example:

```
create table d1 ( c1 date );
load table d1 ( c1 binary with null byte ) from 'filename' quotes off
escapes off format binary;
```

If the content of the load input file is 000b32cb00000b32cc00, two rows are loaded to the table. The first row is May 7, 2009 and the second May 8, 2009. A NULL byte is added to the input file after each binary date. If you want NULL loaded into the first row, change the value of the NULL byte in the input file to “x01”.

```
000b32cb01000b32cc00
```

As another example, to load the value 32769 into a `NUMERIC`(20) column, the input file contains:

```
0x0102 0x5000 0x0ad1 0x0003 0x0000 0x00
```

This includes the NULL byte.

To load 23456789012345678.12 into a column defined as `NUMERIC`(19,2), the load input file contains:

```
0x0106 0x4f00 0x04b0 0x162e 0x04d2 0x1ed2 0x0d80 0x0002 0x0000 0x00
```

The digits are followed by the NULL BYTE (0x00).

There are seven (numbered 0 – 6) unsigned shorts in the digits array of the structure that represents this numeric quantity. “digits[0]” contains the least-significant digits.

```
digits[0] = 0x04b0 (decimal 120)
digits[1] = 0x162e (decimal 5678)
digits[2] = 0x04d2 (decimal 1234)
digits[3] = 0x1ed2 (decimal 7890)
digits[4] = 0x0d80 (decimal 3456)
digits[5] = 0x0002 (decimal 2)
digits[6] = 0x0000
```

The NULL portion of the column specification indicates how to treat certain input values as NULL values, when loading into the table column. These characters can include BLANKS, ZEROS, or any other list of literals you define. When you specify a NULL value or read a NULL value from the source file, the destination column must be able to contain NULLs.

ZEROS is interpreted as follows:

- The column is set to NULL if the input data is entirely binary zeros (as opposed to character zeros).
- If the input data is character zero:
  - NULL(ZEROS) never causes the column to be NULL.
  - NULL('0') causes the column to be NULL. For example:

    Load:

    ```
    CREATE TABLE t1 ( c1 INT, c2 INT );
    ```

    View the input data file, which uses big-endian byte ordering:

    ```
    od -x data.inp
    3030 3030 0000 04d2
    ```

    Execute:

    ```
    LOAD TABLE t1 ( c1 ASCII(4) NULL( '0000' ),
        c2 BINARY )
    FROM 'data.inp'
        FORMAT BINARY
        QUOTES OFF
        ESCAPES OFF;
    ```

    The results:

    ```
    SELECT * FROM t1;
    c1     c2
    NULL   1234
    ```
- If the input data is binary zero (all bits clear):
  - NULL(ZEROS) causes the column to be NULL.
  - NULL('0') never causes the column to be NULL, for example:

    Load:

    ```
    CREATE TABLE t1 ( c1 INT, C2 INT );
    ```

    VIEW the input data file, which uses big-endian byte ordering:

```
od -x data.inp
0000 0000 0000 04d2
```

Execute:

```
LOAD TABLE t1 ( c1 ASCII(4) NULL( zeros ),
    c2 BINARY )
FROM 'data.inp'
    FORMAT BINARY
    QUOTES OFF
    ESCAPES OFF;
```

The results:

```
SELECT * FROM T1;
c1      c2
NULL    1234
```

As another example, if your **LOAD TABLE** statement includes `col1 date( 'yymmdd' ) null(zeros)` and the data to load is 000000, you receive an error indicating that 000000 cannot be converted to a `DATE`(4). To get **LOAD TABLE** to insert a NULL value in `col1` when the data is 000000, either write the NULL clause as `null('000000')`, or modify the data to equal binary zeros and use NULL(ZEROS).

Another way to load NULLs during a binary load is not to supply data for the column in the **LOAD TABLE** statement, if the destination column accepts null values. For example:

```
CREATE TABLE t1 ( c1 INT, c2 INT );
LOAD TABLE T1 ( c2 BINARY ) FROM 'data.inp'
    FORMAT BINARY
    QUOTES OFF
    ESCAPES OFF;

SELECT * FROM T1;
c1      c2
NULL    1234
NULL    1234
```

View the input data file, which uses big-endian byte ordering:

```
od -x data.inp
0000 04d2 0000 04d2
```

Using the INSERT Statement

The **INSERT** statement allows you to insert data without first putting it into a flat file.

Using this command, you can either:

- Insert a specified set of values row by row
- Insert directly from database tables

See the sections that follow for details of these two forms of the command.

See also

- *Interactive Data Imports* on page 272

Inserting Specified Values Row by Row

To add specified values to a table row by row, use Syntax 1 for the **INSERT** statement.

Sybase IQ inserts the first value you specify into the first column you specify, the second value you specify into the second column, and so on. If you omit the list of column names, the values are inserted into the table columns in the order in which the columns were created (the same order as **SELECT *** would retrieve). Sybase IQ inserts the row into the table wherever room is available.

Values can be NULL, any positive or negative number, or a literal.

- Enclose values for `CHAR`, `VARCHAR`, `DATE`, `TIME`, and `TIMESTAMP` or `DATETIME` columns in single or double quotation marks. To indicate a value with a quotation in it use a different set of quotes for the outer quote, such as “Smith' s”.
- For `DATE`, `TIME`, and `TIMESTAMP` or `DATETIME` columns, you must use a specific format.

Note: The `TIMESTAMP` and `DATETIME` data types are identical.

Allowing NULL Values

When you specify values for only some of the columns in a row, NULL is inserted for columns with no value specified, if the column allows NULL.

If you specify a NULL value, the destination column must allow NULLs, or the INSERT is rejected and an error message is produced in the message log. Sybase IQ columns allow NULLs by default, but you can alter this by specifying **NOT NULL** on the column definition in the **CREATE TABLE** statement or in other ways, such as using a primary key, for example.

Example

The following example adds 1995-06-09 into the `l_shipdate` column and 123 into the `l_orderkey` column in the `lineitem` table.

```
INSERT INTO lineitem
    (l_shipdate, l_orderkey)
VALUES('1995-06-09', 123)
```

If you are inserting more than a small number of data rows, it is more efficient to insert selected rows directly from a database, as described in the next section, or to load data from a flat file with the **LOAD TABLE** statement, than to insert values row by row. Consider using a select statement with a few unions instead of inserting values for a few rows, because this requires only a single trip to the server.

See also

- *Data Conversion on Insertion* on page 276

Inserting Selected Rows from the Database

You can insert any number of rows of data, based on the results of a general **SELECT** statement.

To insert data from other tables in the current database, or from a database that is defined as a Specialty Data Store to Sybase IQ, use Syntax 2 for the **INSERT** statement.

For maximum efficiency, insert as many rows as possible in one **INSERT** statement. To insert additional sets of rows after the first insert, use additional **INSERT** statements.

Like other SQL databases, Sybase IQ inserts data by matching the order in which columns are specified in the destination column list and the select list; that is, data from the first column in the select list is inserted into the first destination column, and so on. For both **INSERT SELECT** and **INSERT VALUES**, if you omit destination column names, Sybase IQ inserts data into columns in the order in which they were created.

The tables you are inserting into must exist in the database you are currently connected to. Sybase IQ inserts the data into all indexes for the destination columns.

The columns in the table in the select-list and in the table must have the same or compatible data types. In other words, the selection's value must be, or must be able to be converted to, the data type of the table's column.

With this form of the **INSERT** statement you can specify any of the insert-load-options.

Example

This example shows an insert from one table, `partsupp`, to another, `lineitem`, within the same database.

The data from the source column `l_quantity` is inserted into the destination column `ps_availqty`.

```
INSERT INTO partsupp(ps_availqty)
SELECT l_quantity FROM lineitem
```

Inserting from a Different Database

You can insert data from tables in any accessible database.

- Tables in either the IQ store or the catalog store of the database you are currently connected to.
- Tables in an Adaptive Server Enterprise database.
- A *proxy table* in your current database, that corresponds to a table in a database on a remote server.

Inserting Data Directly from Adaptive Server Enterprise

To insert data from an Adaptive Server Enterprise or SQL Server database, use the **LOCATION** syntax of the **INSERT** statement.

To use insert data directly from Adaptive Server, all of the following must be true:

- The Sybase connectivity libraries must be installed on your system, and the load library path environment variable for your platform must point to them.
- The Adaptive Server Enterprise server to which you are connecting must exist in the `interfaces` file on the local machine.
- You must have read permission on the source ASE or Sybase IQ database, and INSERT permission on the target Sybase IQ database

1. Connect to both the Adaptive Server Enterprise and the Sybase IQ database using the same user ID and password.
2. On the Sybase IQ database, issue a statement using this syntax:

```
INSERT INTO iq_table
LOCATION 'ase_servername.ase_dbname'
{ SELECT col1, col2, col3,...
FROM owner.ase_table }
```

3. Issue a **COMMIT** to commit the insert.

When Sybase IQ connects to the remote server, **INSERT...LOCATION** can also use the remote login for the user ID of the current connection, if a remote login has been created with **CREATE EXTERNLOGIN** and the remote server has been defined with a **CREATE SERVER** statement.

Note: You can also use this method to move selected columns between two Sybase IQ databases.

Loading ASE Text and Images

Sybase IQ does not support the Adaptive Server Enterprise data type `TEXT`, but you can execute **INSERT...LOCATION** (Syntax 3) from both an IQ `CHAR` or `VARCHAR` column whose length is greater than 255 bytes, or a `LONG VARCHAR` column, and from an ASE database column of data type `TEXT`. ASE `TEXT` and `IMAGE` columns can be inserted into columns of other Sybase IQ data types, if Sybase IQ supports the internal conversion. Also note that **INSERT...LOCATION** does not support the use of variables in the **SELECT** statement. By default, if a remote data column contains over 2GB, Sybase IQ silently truncates the column value to 2GB.

Users must be specifically licensed to use the Unstructured Data Analytics functionality. For details, see *Unstructured Data Analytics in Sybase IQ*.

You may substitute curly braces {} for the single quotation marks that delimit the SELECT statement. (Note that curly braces represent the start and end of an escape sequence in the ODBC standard, and may generate errors in the context of ODBC.)

Example

The following command inserts data from the `l_shipdate` and `l_orderkey` columns of the `lineitem` table from the Sybase IQ database `iq11db.dba` on the server `detroit`, into the corresponding columns of the `lineitem` table in the current database.

```
INSERT INTO lineitem
    (l_shipdate, l_orderkey)
     LOCATION 'detroit.iq11db'
    { SELECT l_shipdate, l_orderkey
    FROM lineitem }
```

- The destination and source columns may have different names.
- The order in which you specify the columns is important, because data from the first source column named is inserted into the first target column named, and so on.
- You can use the predicates of the **SELECT** statement within the **INSERT** command to insert data from only certain rows in the table.

Example

This example inserts the same columns as the previous example, but only for the rows where the value of `l_orderkey` is 1. Also in this example, the TDS packet size is specified as 512 bytes.

```
INSERT INTO lineitem
    (l_shipdate, l_orderkey)
    LOCATION 'detroit.iqdb'
    PACKETSIZE 512
    { SELECT l_shipdate, l_orderkey
    FROM lineitem
    WHERE l_orderkey = 1 }
```

See also

- *Bulk Loads with the LOAD TABLE Statement* on page 254

Interactive Data Imports

If you are inserting small quantities of data, you may prefer to enter it interactively through Interactive SQL, using the **INSERT** statement.

For example, you can insert listed values a single row at a time with the following command:

```
INSERT INTO T1
VALUES ( ... )
```

Note: Do not use the Import option on the Interactive SQL Data menu. It is not supported for use with Sybase IQ databases.

See also

- *Using the INSERT Statement* on page 268

Moving Data Between Systems with Different Endian Formats

You can move data from a database in big-endian format to a database in little-endian format.

Prerequisites

Note: Before you begin, make sure that you have a process for capturing your database and table schema.

The following example loads a table named `lineitem` and identifies one extract file on UFS (file system) called `lineitem_binary.inp`.

Check operating system documentation for the maximum file size for your system. For example, an extract file on Sun Solaris x64 has a maximum size of 512GB.

Task

This procedure moves table definitions but does not include migration of database objects, such as stored procedures or events, which you must re-create.

For example, Sybase IQ databases built on Sun64 SPARC systems store binary data in big-endian (most significant byte first) format. Because Sun Solaris x64 is a little-endian system, you cannot upgrade Sybase IQ databases built on Sun64 SPARC with **ALTER DATABASE UPGRADE** to run on Sun Solaris x64 systems.

To move data for each database across hardware platforms of different endian structures, you must:

- Copy the database schema from the source platform (tables, indexes, and so on).
- Create a new database on the target platform.
- Perform a binary data dump from the source database.
- Load data into the new target database.

1. Activate the extract utility:

   ```
   SET TEMPORARY OPTION Temp_Extract_Name1 =
   'lineitem_binary.inp'
   ```

   ```
   SET TEMPORARY OPTION Temp_Extract_Name2 = ''
   ```

2. Set up a binary extract of the `lineitem` table:

   ```
   SET TEMPORARY OPTION Temp_Extract_Binary = 'on'
   ```

   ```
   SET TEMPORARY OPTION Temp_Extract_Swap = 'off'
   ```

3. Place output in the file `lineitem_binary.inp`:

```
SELECT * FROM lineitem
```

4. Turn off the extract utility:

```
SET TEMPORARY OPTION Temp_Extract_Name1 = ''
```

5. Create a duplicate of your database on the target system.
6. Assuming table `lineitem` as defined below, load the `lineitem` table as follows:

```
LOAD TABLE lineitem
( l_orderkey      BINARY WITH NULL BYTE,
l_partkey         BINARY WITH NULL BYTE,
l_suppkey         BINARY WITH NULL BYTE,
l_linenumber      BINARY WITH NULL BYTE,
l_quantity        BINARY WITH NULL BYTE,
l_extendedprice   BINARY WITH NULL BYTE,
l_discount        BINARY WITH NULL BYTE,
l_tax             BINARY WITH NULL BYTE,
l_returnflag      BINARY WITH NULL BYTE,
l_linestatus      BINARY WITH NULL BYTE,
l_shipdate        BINARY WITH NULL BYTE,
l_commitdate      BINARY WITH NULL BYTE,
l_receiptdate     BINARY WITH NULL BYTE,
l_shipinstruct    BINARY WITH NULL BYTE,
l_shipmode        BINARY WITH NULL BYTE,
l_comment         BINARY WITH NULL BYTE )
FROM 'C:\\mydata\\lineitem_binary.inp'
FORMAT BINARY
STRIP OFF
QUOTES OFF
ESCAPES OFF
PREVIEW ON
BYTE ORDER HIGH;
COMMIT
```

Note particularly two clauses:

- BINARY WITH NULL BYTE is required when loading a binary file.
- BYTE ORDER HIGH specifies the byte order from the system where the data *originated*. The source database in this example is a big-endian platform; therefore, this data requires byte order HIGH. (Little-endian databases require byte order LOW.)

When loading a multiplex database, use absolute (fully qualified) paths in all file names. Do not use relative path names.

Inserting into Tables of a Join Index

You load or insert data into the tables underlying a join index, just as you would any other indexes. There is only one difference: the data in a join index must be synchronized before you can use the join index to resolve queries.

Note: You cannot update a base table that is part of any join index. You can only insert, load, or delete.

When you first create a join index, Sybase IQ synchronizes the join index for you automatically. It does not matter whether you create the join index before or after loading. The order also does not affect performance of the load or synchronization.

Once you have created a join index, however, if you insert or load data into any of its underlying tables except the top table in the join hierarchy, you must synchronize it explicitly. To do so, use the **SYNCHRONIZE JOIN INDEX** statement. See *Reference: Statements and Options*.

Once any user has updated any of the tables in a join index, no other user can update any of the tables underlying that join index until the join index has been synchronized.

Updating from Different Connections May Cause Errors

When more than one user inserts into or deletes from different tables that participate in the same join index, the second user's update will fail unless the synchronize commits before the second user's transaction starts. This failure occurs if either of the following conditions exist:

- The second user's transaction begins before the first user's transaction commits.
- The second user tries to update after the first user's transaction commits, but before the join index is synchronized.

This problem occurs because Sybase IQ makes a new version of the join index when any of its underlying tables is updated. The new version is not visible to other transactions that have already begun. The problem does not occur when one user makes all of the changes, because the newer table version is visible to the user who made the original changes.

For example, assume that tables A, B, and C are all members of the same join index. User 2 begins a transaction, and writes to another table not involved in the join. Now, User 1 inserts into table B. This action creates a new version of table B, and a new version of the join index. User 2 then tries to write to table C. Even though no other user has changed table C, because C is a member of the join index it can't be updated until the join index is synchronized.

Insertions into Primary and Foreign Key Columns

You load or insert data into primary key and foreign key columns just as you would into any other column.

When you insert into a primary key, Sybase IQ checks that each value is unique. If it is not, an error occurs.

Load or Extraction of Large Object Data

Users must be specifically licensed to use the Unstructured Data Analytics functionality.

To load or extract large object data, see *Unstructured Data Analytics in Sybase IQ*.

Data Conversion on Insertion

The data you enter into your Sybase IQ database will likely come from diverse sources.

Not all of your data will match the Sybase IQ data types exactly. Some of it will need to be converted. Data is converted in two ways: explicitly and implicitly. For example, to insert `INT` data into a `CHAR` column you must convert it explicitly.

Implicit conversions can occur:

- When you insert data selected from another column in the same database
- When you insert data selected from another database
- When you load data from a flat file

When an explicit conversion is needed, the way that you specify the conversion depends on whether you are loading from a flat file or inserting selected rows:

- In the **LOAD TABLE** statement, you convert data explicitly by specifying a format in the *column-spec*.
- In the **INSERT** statement, you convert data explicitly using the data conversion functions **CAST**, **CONVERT**, and **DATEPART** in the **SELECT** statement or **VALUES** list.

While most Sybase IQ data types are fully compatible with SQL Anywhere and Adaptive Server Enterprise data types of the same name, there are some differences.

For compatibility among versions, a few data types have been defined as synonyms of other data types:

- `DECIMAL` is a synonym for `NUMERIC`.
- `INTEGER` is a synonym for `INT`.
- `DATETIME` is a synonym for `TIMESTAMP`.
- `FLOAT` (*precision*) is a synonym for `REAL` or `DOUBLE`, depending on the value of *precision*. For Adaptive Server Enterprise, `REAL` is used for *precision* less than or equal to 15, and `DOUBLE` for *precision* greater than 15. For Sybase IQ and SQL Anywhere, the cutoff is platform-dependent, but on all platforms the cutoff value is greater than 22.
- `MONEY` is an Adaptive Server Enterprise-compatible synonym for `NUMERIC`(19,4), allowing NULL.
- `SMALLMONEY` is an Adaptive Server Enterprise-compatible synonym for `NUMERIC`(10,4), allowing NULL.

You can use a synonym interchangeably with its standard data type. Data is stored internally as the standard data type, where synonyms exist. In error messages, the standard name appears in place of the synonym.

Note: By default, Sybase IQ assumes that input data is binary (numeric data) and tries to insert it that way. However, this presumes that the input column length in bytes must match the destination column length in bytes. If not, the insert will fail or lead to unexpected results. For example, if you attempt to insert an input column with integer data of 4 bytes into a SMALLINT destination column, Sybase IQ loads only the first 2 bytes of that input column.

See also

- *Inserting Specified Values Row by Row* on page 269
- *Explicit Data Conversions in IQ* on page 278
- *Matching Adaptive Server Enterprise Data Types* on page 291

Data from Pre-Version 15 Sybase IQ

If you are moving data into Sybase IQ version 15.x from an earlier version, you must convert certain data types before inserting or loading them.

See "Migrating Data from Prior Versions" in the *Installation and Configuration Guide*.

Load Conversion Options

The following table lists the conversion options for the **LOAD TABLE** statement in alphabetical order and gives a brief description of what each option does.

Table 24. Conversion options for loading from flat files

Option	Sybase IQ Data types	Action
ASCII	TINYINT, SMALLINT, INT (or INTEGER), UNSIGNED INT, BIGINT, UNSIGNED BIGINT, NUMERIC (or DECIMAL), REAL, DOUBLE, BIT, DATE, TIME, TIMESTAMP (or DATETIME)	By default, Sybase IQ assumes input data is binary of appropriate width for the data type. Using **ASCII** allows you to tell Sybase IQ that data is in character format and lets you specify how wide it is. This option allows E notation for REAL data, but it can hinder your performance.
ASCII	CHAR, VARCHAR	By default, Sybase IQ assumes same column width between source and destination columns, which may cause it to read input file incorrectly. This option lets you specify a different width for the input column.
DATE	DATE	Converts ASCII date input of a fixed format to binary.
DATETIME	TIMESTAMP (or DATETIME) or TIME	Converts ASCII time or date/time input of a fixed format to binary. The input specification is based on either a 12-hour or 24-hour clock.

Option	Sybase IQ Data types	Action
TIME	`TIME`	Converts ASCII time input of a fixed format to binary.
NULL	all	Lets you specify which input data values to convert to NULL on insert.

Note: When loading from a flat file, use binary data if you have a choice of using binary or character data. Using binary input can improve performance by eliminating conversion costs.

See also

- *Loads that Specify Input Data Format* on page 255

Explicit Data Conversions in IQ

When you use the **INSERT** statement to insert data directly from a database rather than from a flat file, you cannot use the load conversion options.

If the data requires explicit conversion, you must use one of the conversion functions, **CAST** or **CONVERT**, in the **SELECT** statement or **VALUES** list where you specify the data to be inserted. If the data is converted implicitly, Sybase IQ handles the conversion automatically.

An implicit or explicit conversion is required whenever data types in a **SELECT** statement need to match, but do not. This occurs when you do an **INSERT SELECT** from one data type to another, but it also occurs whenever you compare or compute values for differing data types.

The following tables show:

- Which conversions Sybase IQ does implicitly (I)
- Which conversions you must do explicitly (E)
- Which conversions are unsupported (U)

These conversions apply to data within a Sybase IQ database, or coming from an SQL Anywhere database, or any other database connected as a Specialty Data Store.

The first table shows implicit (I), explicit (E), and unsupported (U) conversions when there is no **WHERE** clause in the **SELECT** statement, or when the **WHERE** clause is based on a comparison operation (=, > or <).

Table 25. IQ conversions for comparison operations

	To:																
From:	**ti**	**si**	**in**	**ui**	**bi**	**ub**	**nu**	**rl**	**dl**	**bt**	**dt**	**tm**	**ts**	**ch**	**vc**	**bn**	**vb**
tinyint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
smallint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I

	To:																
From:	**ti**	**si**	**in**	**ui**	**bi**	**ub**	**nu**	**rl**	**dl**	**bt**	**dt**	**tm**	**ts**	**ch**	**vc**	**bn**	**vb**
int	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
unsigned int	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
bigint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
unsigned bigint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
numeric	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	U	U
real	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	U	U
double	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	U	U
bit	I	I	I	I	I	I	I	I	I	I	U	U	U	I	I	I	I
date	E	E	E	E	E	E	E	E	E	U	I	U	I	E	E	U	U
time	E	E	E	E	E	E	E	E	E	U	U	I	E	E	E	U	U
time-stamp	E	E	E	E	E	E	E	E	E	U	E	I	I	E	E	U	U
char	E	E	E	E	E	E	E	E	E	I	E	E	E	I	I	I	I
varchar	E	E	E	E	E	E	E	E	E	I	E	E	E	I	I	I	I
binary	I	I	I	I	I	I	U	U	U	U	U	U	U	I	I	I	I
varbinary	I	I	I	I	I	I	U	U	U	U	U	U	U	I	I	I	I

The following list contains the descriptions of the codes used in the tables:

Code	Data type	Code	Data type	Code	Data type
ti	tinyint	nu	numeric	ts	timestamp
si	smallint	rl	real	ch	char
in	int	dl	double	vc	varchar
ui	unsigned int	bt	bit	bn	binary
bi	bigint	dt	date	vb	varbinary
ub	unsigned bigint	tm	time		

The second table shows implicit (I), explicit (E), and unsupported (U) conversions when the **WHERE** clause in a **SELECT** statement is based on an arithmetic operation (+, –, etc.).

Table 26. IQ conversions for arithmetic operations

	To:																
From:	**ti**	**si**	**in**	**ui**	**bi**	**ub**	**nu**	**rl**	**dl**	**bt**	**dt**	**tm**	**ts**	**ch**	**vc**	**bn**	**vb**
tinyint	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	I	I
smallint	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	I	I
int	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	I	I
unsigned int	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	I	I
bigint	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	I	I
unsigned bigint	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	I	I
numeric	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
real	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
double	I	I	I	I	I	I	I	I	I	I	U	U	U	E	E	U	U
bit	I	I	I	I	I	I	I	I	I	I	U	U	U	I	I	I	I
date	U	U	U	U	U	U	U	U	U	U	U	I	U	U	U	U	U
time	U	U	U	U	U	U	U	U	U	U	I	U	U	U	U	U	U
time-stamp	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U	U
char	E	E	E	E	E	E	E	E	E	I	U	U	U	I	I	I	I
varchar	E	E	E	E	E	E	E	E	E	I	U	U	U	I	I	I	I
binary	I	I	I	I	I	I	U	U	U	U	U	U	U	I	I	I	I
varbinary	I	I	I	I	I	I	U	U	U	U	U	U	U	I	I	I	I

Note: In arithmetic operations, `bit` data is implicitly converted to `tinyint`.

The third table shows implicit (I), explicit (E), and unsupported (U) conversions for the **INSERT** and **UPDATE** statements.

Table 27. IQ conversions for INSERT and UPDATE

	To:																
From:	**ti**	**si**	**in**	**ui**	**bi**	**ub**	**nu**	**rl**	**dl**	**bt**	**dt**	**tm**	**ts**	**ch**	**vc**	**bn**	**vb**
tinyint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
smallint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
int	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
unsigned int	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
bigint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
unsigned bigint	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	I	I
numeric	I	I	I	I	I	I	I	I	I	E	E	E	E	E	E	U	U
real	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	U	U
double	I	I	I	I	I	I	I	I	I	I	E	E	E	E	E	U	U
bit	I	I	I	I	I	I	I	I	I	I	U	U	U	I	I	I	I
date	E	E	E	E	E	E	E	E	E	E	I	U	I	E	E	U	U
time	E	E	E	E	E	E	E	E	E	E	U	I	E	E	E	U	U
time-stamp	E	E	E	E	E	E	E	E	E	E	E	I	I	E	E	U	U
char	I	I	I	I	I	I	I	I	I	I	E	E	E	I	I	I	I
varchar	I	I	I	I	I	I	I	I	I	I	E	E	E	I	I	I	I
binary	I	I	I	I	I	I	U	U	U	I	U	U	U	I	I	I	I
varbinary	I	I	I	I	I	I	U	U	U	I	U	U	U	I	I	I	I

See also

- *Data Conversion on Insertion* on page 276
- *Matching Adaptive Server Enterprise Data Types* on page 291
- *Supported Column Default Values* on page 346

Column Width Issues

Sybase IQ assumes the width of the input data is the same as the destination column width and reads the input file accordingly.

If they are not the same width, Sybase IQ may read too few or too many bytes of the input file for that column. The result is that the read for that column may be incorrect, and the reads for subsequent columns in the input file will be incorrect, because they will not start at the correct position in the input file.

For example, if `input_column1` is 15 bytes wide and `destination_column1` is 10 bytes wide, and you do not specify the **ASCII** conversion option, Sybase IQ assumes the input column is only 10 bytes wide. This is fine for `destination_column1`, because the input data is truncated to 10 bytes in any case. But it also means that Sybase IQ assumes that the next column in the input file starts at byte 11, which is still in the middle of the first column, instead of at byte 16, which is the correct starting position of the next column.

Conversely, if `input_column1` is 10 bytes wide and `destination_column1` is 15 bytes wide, and you do not specify the **ASCII** conversion option, Sybase IQ assumes the input column is 15 bytes wide. This means that Sybase IQ reads all of `input_column1` plus 5 bytes into the next column in the input file and inserts this value into `destination_column1`. So, the value inserts into `destination_column1` and all subsequent columns are incorrect.

To prevent such problems, use the **ASCII** conversion option. With this option, Sybase IQ provides several ways to specify the fixed or variable width of an input column. Your input data can contain fixed-width input columns with a specific size in bytes, variable width input columns with column delimiters, and variable width input columns defined by binary prefix bytes.

Faster Date and Time Loads

Sybase IQ has performance optimizations built in for ascii-to-binary conversions on date, time, and datetime data during loads. If the raw data you are loading exactly matches one of these formats, you can significantly decrease load time by using the appropriate format.

The recognized formats are:

- "YYYY-MM-DD"
- "YYYY/MM/DD"
- "YYYY.MM.DD"
- "YYYYMMDD"
- "MM-DD-YYYY"
- "MM/DD/YYYY"
- "DD-MM-YYYY"
- "DD/MM/YYYY"
- "DD.MM.YYYY"
- "HH:NN:SS"
- "HHNNSS"
- "HH:NN:SS.S"
- "HH:NN:SS.SS"
- "HH:NN:SS.SSS"

- "HH:NN:SS.SSSS"
- "HH:NN:SS.SSSSS"
- "HH:NN:SS.SSSSSS"
- "YYYY-MM-DD HH:NN:SS"
- "YYYYMMDD HHNNSS"
- "YYYY-MM-DD HH:NN:SS.S"
- "YYYY-MM-DD HH:NN:SS.SS"
- "YYYY-MM-DD HH:NN:SS.SSS"
- "YYYY-MM-DD HH:NN:SS.SSSS"
- "YYYY-MM-DD HH:NN:SS.SSSSS"
- "YYYY-MM-DD HH:NN:SS.SSSSSS"

When you load a table having one or more date, time, or datetime columns *and* the input format is in one of the above formats, then the load can run significantly faster if you explicitly specify the appropriate format on the load statement. Otherwise, the load can run very slowly.

Suppose that your table had a date column, created as follows:

```
CREATE TABLE table1(c1 DATE);
```

To load the table, use a statement like this:

```
 LOAD TABLE table1 (c1 ASCII(10)) FROM ...
```

If the raw data format is in a format that has been optimized (such as YYYY-MM-DD), the load will be much faster.

The following sections describe the conversion options in greater detail.

ASCII Input Conversion

Convert ASCII input data to binary.

Use the **ASCII** conversion option to either:

- Convert ASCII input data to binary and specify the width of the input column so data can be read in correctly for that column, or
- Insert ASCII data into an `ASCII` data type column when the width of the input column is different from the width of the destination column. This option lets you specify how much of the input data it should read for each column.

You can use this option with any of the Sybase IQ data types, with 1, 2, or 4 prefix bytes, and with a column delimiter.

Truncation of Data for VARCHAR and CHAR Columns

If the width of the input column is greater than the width of the destination column, Sybase IQ truncates the data upon insertion.

If the width of the input data is less than the width of the destination column, for `CHAR` or `VARCHAR` data types Sybase IQ pads the data with spaces in the table upon insertion.

Variable width inserts to a `VARCHAR` column will not have trailing blanks trimmed, while fixed-width inserts to a `VARCHAR` column will be trimmed. For example, assume that you are inserting into column `varcolumn` in a table called `vartable`. The following would constitute a fixed-width insert, where the value would not be trimmed because you explicitly say to include the two blanks (indicated by __ here):

```
INSERT INTO vartable VALUES ('box__')
```

If instead you inserted the same value from a flat file using delimited input, it would be a variable-width insert, and the trailing blanks would be trimmed.

The following table illustrates how the **ASCII** conversion option works with the Sybase IQ data types. The example inserts the data from the flat ASCII file `shipinfo.t` into the Sybase IQ table `lineitem` and summarizes the content and format of the input data and the table.

Table 28. Input file conversion example

File *shipinfo.t*			Table *lineitem*		
Column	***Format***	***Width***	***Column***	***Data type***	***Width***
l_shipmode	CHAR	15	l_shipmode	VARCHAR	30
l_quantity	ASCII	8	l_quantity	INT	4

For the `l_shipmode` column, you insert ASCII data into an ASCII column (that has a VARCHAR data type). Notice the width of the two columns is different. In order for the insert on this column and the subsequent `l_quantity` column to be correct, you specify the width of the `l_shipmode` column so the correct amount of input data is read at the correct position.

For the `l_quantity` column, you are inserting ASCII data into a binary column (**INT** data type). In order for the insert on this column to be correct, you must convert the input data into binary and indicate the width of the input column.

The command for this is shown in the following UNIX example.

```
LOAD TABLE lineitem(
    l_shipmode ASCII(15),
    l_quantity ASCII(8),
FILLER(1))
FROM '/d1/MILL1/shipinfo.t'
PREVIEW ON
```

Substitution of NULL or Blank Characters

Sybase IQ supports zero-length VARCHAR data. If the length of a VARCHAR cell is zero and the cell is not NULL, you get a zero-length cell.

For all other data types, if the length of the cell is zero, Sybase IQ inserts a NULL.

This treatment of zero-length character data is ANSI behavior. If you require non-ANSI behavior, see the **NON_ANSI_NULL_VARCHAR** option in *Reference: Statements and Options*.

The DATE Option

Use the **DATE** conversion option to insert ASCII data that is stored in a fixed format into a `DATE` column.

This option converts the ASCII data input to binary and specifies the format of the input data. (The **DATE** format is used internally to interpret the input; it does not affect the storage or output format of the data.) See the **ASCII** conversion format for more information.

Example

In this Windows example, data for the `l_shipdate` column is converted from the specified format into binary. The 1-byte **FILLER** skips over carriage returns in the input file.

```
LOAD TABLE lineitem(
    l_orderkey NULL(ZEROS) ASCII(4),
    l_partkey ASCII(3),
    l_shipdate DATE('MM/DD/YY'),
    l_suppkey ASCII(5),
FILLER(1))
FROM 'C:\\MILL1\\shipinfo.t'
PREVIEW ON
```

DATE Formats

Specify the format of the input data using y or Y for years, m or M for months, d or D for days, and j or J for Julian days.

The length of the format string is the width of the input column.

Table 29. Formatting dates

Option	Meaning
yyyy or YYYY yy or YY	Represents number of year. Default is 1900.
mm or MM	Represents number of month. Always use leading zeros for number of the month where appropriate, for example '05' for May. If you omit the month from a `DATE` value, the day is treated as a Julian date. If you enter only the month, for example, '03', Sybase IQ applies the default year and day and converts it to '1900-03-01'.
dd or DD jjj or JJJ	Represents number of day. Default day is 01. Always use leading zeros for number of day where appropriate, for example '01' for first day. J or j indicates a Julian day (1 to 366) of the year.

On input, the case the format code is ignored.

On output, the case of the format code has the following effect:

- Mixed case (for example, “Dd”) means do not pad with zeros.
- Same case (for example, “DD” or “dd” means do pad with zeros.

For example, a time as 17:23:03.774 using the default time format, but as 17:23:3.774 using 'HH:NN:Ss.SSS'.

The next table shows examples of how date input data looks and how to specify the format with the **DATE** conversion option. Following the table are general rules for specifying dates.

Table 30. Sample DATE format options

Input Data	Format Specification
12/31/09	**DATE** ('MM/DD/YY')
12-31-09	**DATE** ('MM-DD-YY')
20091231	**DATE** ('YYYYMMDD')
12/09	**DATE** ('MM/YY')
2009/123	**DATE** ('YYYY/JJJ')

- The **DATE** specification must be in parentheses and enclosed in single or double quotes.
- Sybase IQ stores only the numbers of the year, month, and day; it does not store any other characters that might appear in the input data. However, if the input data contains other characters, for example, slashes (/), dashes (-), or blanks to separate the month, day, and year, the **DATE** format must show where those characters appear so they can be ignored.
- Use any character other than Y, M, J, or D to indicate the separator character you want Sybase IQ to skip over. You can even use blanks.
- If a **DATE** format includes only a year and a day number within the year, Sybase IQ treats the date as a Julian date. For example, 2009-33 is the 33rd day in the year 2009, or February 2, 2009.
- If a year is specified with only two digits, for example “5/27/32”, then Sybase IQ converts it to 19yy or 20yy, depending on the year and on the setting of the **NEAREST_CENTURY** option.

NEAREST_CENTURY setting	Year specified as	Years assumed
Default (50)	00-49 50-99	2000-2049 1950-1999
0	any	1900s
100	any	2000s

See also

- *Specifying the Format for DATETIME Conversions* on page 288

The DATETIME Conversion Option

Use the **DATETIME** conversion option to insert ASCII data that is stored in a fixed format into a `TIME` or `TIMESTAMP` or `DATETIME` column.

This option converts the ASCII data input to binary and specifies the format of the input data. (The **DATETIME** format is used internally to interpret the input; it does not affect the storage or output format of the data.) See the **ASCII** conversion format for more information.

Note: For compatibility with previous releases, you can specify that a column contains `DATETIME` data. However, such data is stored internally as the equivalent format, `TIMESTAMP`.

Here is the syntax:

```
DATETIME ('input-datetime-format')
```

In this UNIX example, slashes are separators in the date portion of the input data, and colons are separators in the time portion:

```
LOAD TABLE lineitem(
    l_quantity ASCII(4),
    l_shipdate DATETIME('MM/DD/YY hh:mm:ss'),
FILLER(1))
FROM '/d1/MILL1/tt.t'
PREVIEW ON
```

In this UNIX example, the FILLER(1) clause prevents Sybase IQ from inserting a NULL in the next column (VWAP) after the DATETIME column:

```
LOAD TABLE snapquote_stats_base
SYMBOL ‘\x09’,
snaptime DATETIME('MM/DD/YY hh:mm:ss'),
FILLER(1))
VWAP ‘\x09’,
RS_DAY ‘\x09’,
FROM '/d1/MILL1/tt.t'
PREVIEW ON
```

In this UNIX example, the destination columns contain `TIME` data, but the input data is `DATETIME`. You use the **DATETIME** conversion option, and use **FILLER** to skip over the date portion.

```
LOAD TABLE Customers(
    open_time DATETIME('hh:mmaa'),
    close_time DATETIME('hh:mmaa'),
FILLER(9))
FROM '/d1/MILL1/tt.t'
PREVIEW ON
```

Specifying the Format for DATETIME Conversions

Specify the format of the `DATETIME` input.

Specify the format using:

- Y or y for years
- M or m for months
- D or d for days
- H or h to indicate hours
- N or n to indicate minutes (mm is also accepted when colons are used as separators
- S or s to indicate seconds and fraction of a second

The length of the format string is the width of the input column. The following table describes the time formatting options.

Table 31. Formatting times

Option	Meaning
hh HH	Represents hour. Hour is based on 24-hour clock. Always use leading zeros for hour where appropriate, for example '01' for 1 am. '00' is also valid value for hour of 12 am.
nn	Represents minute. Always use leading zeros for minute where appropriate, for example '08' for 8 minutes.
ss[.ssssss]	Represents seconds and fraction of a second.
aa	Represents the a.m. or p.m designation.
pp	Represents the p.m designation only if needed. (This is incompatible with Sybase IQ releases prior to 12.0; previously, pp was synonymous with aa.)
hh	Sybase IQ assumes zero for minutes and seconds. For example, if the `DATETIME` value you enter is '03', Sybase IQ converts it to '03:00:00.0000'.
hh:nn or hh:mm	Sybase IQ assumes zero for seconds. For example, if the time value you enter is '03:25', Sybase IQ converts it to '03:25:00.0000'.

The following table shows examples of how time input data may look and how to specify the format for the **DATETIME** option. Following this table are the general rules for specifying times.

Table 32. DATETIME format options

Input Data	Format Specification
12/31/00 14:01:50	**DATETIME** ('MM/DD/YY hh:nn:ss')

Input Data	Format Specification
123100140150	**DATETIME** ('MMDDYYhhnnss')
14:01:50 12-31-00	**DATETIME** ('hh:mm:ss MM-DD-YY')
12/31/00 14:01:12.456	**DATETIME** ('MM/DD/YY hh:nn:sssssss')
12/31/00 14:01:.123456	**DATETIME** ('MM/DD/YY hh:mm:sssssss')
12/31/00 02:01:50AM	**DATETIME** ('MM/DD/YY hh:mm:ssaa')
12/31/00 02:01:50pm	**DATETIME** ('MM/DD/YY hh:mm:sspp')

- Specification letters for time components must be in enclosed in parentheses and single or double quotation marks.
- The input data can include up to nine positions for seconds, including a floating decimal point, to allow for fractional seconds. On input and query, the decimal point floats, so you can specify up to six decimal positions. However, Sybase IQ always stores only six decimal positions with two positions for whole seconds (ss.ssssss). Any more decimal positions are not permitted.
- Separators are used between the time elements. You can use any character as a separator, including blanks. The example uses ':' (colons).
- Sybase IQ stores only the numbers of hours, minutes, and seconds; it does not store any other characters which might appear in the input data. However, if the data contains other characters, for example colons (:) or blanks to separate hours, minutes, and seconds, the time portion of the format specification must show where those characters appear so that Sybase IQ knows to skip over them.
- To indicate whether a particular value is a.m. or p.m., the input data must contain an upper- or lowercase 'a' or 'p' in a consistent place. To indicate where Sybase IQ should look for the a.m. or p.m. designation, put a lowercase only 'aa' or 'pp' in the appropriate place in the format specification. `aa' specifies a.m./p.m. is always indicated, while `pp' specifies that pm is indicated only if needed.
- The format specification must have a character to match every character in the input; you cannot have an 'm' in the format specification to match the 'm' in the input, because 'm' is already used to indicate minutes.
- In the time section, when hours or minutes or seconds are not specified, Sybase IQ assumes 0 for each.

See also

- *DATE Formats* on page 285

NULL Data Conversions

Use the **NULL** conversion option to convert specific values in the input data to NULLs when inserting into Sybase IQ column indexes.

This option can be used with any columns, but the column must allow NULLs. You can specify this conversion option with any Sybase IQ data type.

This is the syntax:

```
NULL ({BLANKS | ZEROS | literal' ['literal']...})
```

where:

- **BLANKS** indicates that blanks convert to NULLs.
- **ZEROS** indicates that binary zeros convert to NULLs.
- `literal` indicates that all occurrences of the specified literal convert to NULLs. The specified literal must match exactly, including leading and/or trailing blanks, with the value in the input file, for Sybase IQ to recognize it as a match. You can list up to 20 literal values.

You may need to use additional conversion options on the same column. For example, to insert ASCII data into an `INT` column, which is stored in binary format, and convert blanks in the input data to NULLs when inserted, use the **ASCII** conversion option to convert the input to binary and the **NULL** conversion option to convert blanks to NULLs.

This is a Windows example:

```
LOAD TABLE lineitem(
    l_orderkey NULL(ZEROS) ASCII(4),
    l_partkey ASCII(3),
    l_shipdate date('MM/DD/YY'),
    l_suppkey ascii(5),
FILLER(1))
FROM 'C:\\MILL1\\tt.t'
PREVIEW ON
```

Other Factors Affecting the Display of Data

Whenever Sybase IQ requires an explicit or implicit conversion from one data type to another during a query or insert, it always truncates the results.

The following describes such situations:

- When you explicitly convert data from a higher scale to a lower scale, Sybase IQ truncates the values in the results. For example, if you **CAST** a column value in a query to a scale 2 when it is stored with a scale 4, values such as 2.4561 become 2.45.
- When Sybase IQ implicitly converts from a higher scale to a lower scale during an insertion, it truncates the values before inserting the data into the table. For example, if you

insert from one table with a data type of `NUMERIC(7,3)` to another table with a data type of `DECIMAL(12,2)`, values such as 2.456 will become 2.45.

- When an arithmetic operation results in a higher scale than the predetermined scale, Sybase IQ truncates the results to fit the scale after it has been determined using the rules defined in *Reference: Building Blocks, Tables, and Procedures > SQL Data Types > Numeric data types*.

If your results require rounding of the values instead of truncation, you should use the **ROUND** function in your command. However, for inserts, the **ROUND** function can only be part of its query expression.

The maximum precision for numeric data is 126.

Matching Adaptive Server Enterprise Data Types

The tables below show which Sybase IQ data types are compatible with Adaptive Server Enterprise data types.

Here are some general rules:

- Sybase IQ character string types accept any Adaptive Server Enterprise character string type.
- Sybase IQ exact numeric types accept any Adaptive Server Enterprise number types. However, if the Sybase IQ data type holds a smaller amount of data than the Adaptive Server Enterprise type, the value converts to a NULL (for example, when inserting data from the underlying database into tables).
- Sybase IQ date/time types accept any Adaptive Server Enterprise date/time types.

See also

- *Data Conversion on Insertion* on page 276
- *Explicit Data Conversions in IQ* on page 278

Unsupported Adaptive Server Enterprise Data Types

Not all Adaptive Server Enterprise data types are supported.

These Adaptive Server Enterprise data types are not supported by Sybase IQ in this version:

- `nchar`, `nvarchar`
- `unichar`, `univarchar`, `unitext`
- `text`
- `image`
- `unsigned smallint`
- `native Java data types`
- `XML data type`

Note the following:

- Sybase IQ supports the Adaptive Server Enterprise text and image types via binary large object (BLOB) and character large object (CLOB) data types. For details, see *Unstructured Data Analytics in Sybase IQ*.
- In Sybase Central, you can create columns with certain data types in the SYSTEM dbspace (catalog store) only. In the Create Table wizard, choose Create a Table in the System Store.
- Sybase IQ does not support the Adaptive Server Enterprise data types `DATE`, `TEXT`, `UNSIGNED SMALLINT`, `NCHAR`, `NVARCHAR`, `UNICHAR`, `UNIVARCHAR`, or `UNITEXT`, but you can insert data from an Adaptive Server Enterprise database column of data type `DATE`, `TEXT`, `UNSIGNED SMALLINT`, `NCHAR`, `NVARCHAR`, `UNICHAR`, `UNIVARCHAR`, or `UNITEXT`, using the **LOCATION** syntax of the **INSERT** statement.

Adaptive Server Enterprise Data Type Equivalents

The table below indicates the Adaptive Server Enterprise exact numeric types and the Sybase IQ equivalents.

Table 33. Integer data types

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`int`	`INT`,`BIGINT`,`UNSIGNED INT`, `UNSIGNED BIGINT`, or `NUMERIC`	Sybase IQ does not allow scaled integers, such as INT(7,3). Data in the form INT(*precision,scale*) is converted to `NUMERIC`(*precision,scale*). This differs from Sybase IQ versions prior to 12.0, and from Adaptive Server Enterprise, in which `int` data types can be values between -2,147,483,648 and 2,147,483,647, inclusive. To handle larger integer values, you can use a `BIGINT`, an unsigned integer (`UNSIGNED INT`), or an `UNSIGNED BIGINT` data type. With `UNSIGNED INT`, the last bit is used as part of the value. There is no positive or negative indication; all numbers are assumed to be positive, so the value can go up to 4,294,967,295.
`numeric`	`DECIMAL` or `NUMERIC` with appropriate precision	If the precision of the Sybase IQ data type you define is too small to store the Adaptive Server Enterprise value, the value converts to NULL.
`decimal`	`DECIMAL` or `NUMERIC` with appropriate precision	See above.

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`smallint`	`SMALLINT` or `NUMERIC`	Sybase IQ `SMALLINT`does not allow precision and scale. Adaptive Server Enterprise `smallint(precision,scale)` is converted to `NUMERIC(precision,scale)`See `INT` above.
`tinyint`	`TINYINT`	Sybase IQ `TINYINT` columns do not allow precision and scale. Adaptive Server Enterprise `tinyint(precision,scale)` is converted to `NUMERIC(precision,scale)`. See `INT` above.
`bit`	`BIT`	
`unsigned smallint`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `unsigned smallint`, but you can insert data from an Adaptive Server Enterprise database column of data type `unsigned smallint` using **INSERT...LOCATION**.

The following table indicates the Adaptive Server Enterprise approximate data types and the Sybase IQ equivalents.

Table 34. Approximate numeric data types

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`float (precision)`	`FLOAT (precision)`	IQ supports greater precision for `FLOAT` **HNG** indexes do not allow `FLOAT`, `REAL`, or `DOUBLE` data.
`double precision`	`DOUBLE`	
`real`	`REAL`	

The following table indicates the Adaptive Server Enterprise character data types and their Sybase IQ equivalents.

Table 35. Character data types

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`char`	`CHAR`	Sybase IQ and Adaptive Server Enterprise character (`char` or `CHAR`) data types are the same except that Sybase IQ can handle NULLs. If you want an Sybase IQ `CHAR` column to exactly match an Adaptive Server Enterprise `char` column, specify Sybase IQ column as NOT NULL. Sybase IQ default allows NULLs. Adaptive Server Enterprise `char` columns that allow NULLs are internally converted to `varchar`.
`varchar`	`VARCHAR`	See `char` notes above.
`nchar`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `nchar`, but you can insert data from an ASE database column of data type `nchar` using **INSERT...LOCATION**.
`nvarchar`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `nvarchar`, but you can insert data from an Adaptive Server Enterprise database column of data type `nvarchar` using **INSERT...LOCATION**.
`text`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `text`, but you can insert data from an Adaptive Server Enterprise database column of data type `text` using **INSERT...LOCATION**.
`unichar`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `unichar`, but you can insert data from an Adaptive Server Enterprise database column of data type `unichar` using **INSERT...LOCATION**.
`univarchar`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `univarchar`, but you can insert data from an Adaptive Server Enterprise database column of data type `univarchar` using **INSERT...LOCATION**.
`unitext`	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type `unitext`, but you can insert data from an Adaptive Server Enterprise database column of data type `unitext` using **INSERT...LOCATION**.

The following table indicates the Adaptive Server Enterprise `money` data types and the Sybase IQ equivalents.

Table 36. Money data types

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`money`	`NUMERIC(19,4)`	`money` data is converted implicitly to `NUMERIC(19,4)`.
`smallmoney`	`NUMERIC(10,4)`	

The following table indicates the Adaptive Server Enterprise `DATE/TIME` data types and the Sybase IQ equivalents.

Table 37. DATE/TIME data types

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`datetime`	`TIMESTAMP` or `DATE` or `TIME`	Adaptive Server Enterprise `datetime` columns maintain date and time of day values in 4 bytes for number of days before or after base date of virtual date 0/0/0000 and 8 bytes for time of day, accurate to within one 1,000,000th of a second. Sybase IQ `TIMESTAMP` (or `DATETIME`) columns maintain date and time of day values in two 4-byte integers: 4 bytes for number of days since 1/1/0 and 4 bytes for time of day, based on 24-hour clock, accurate to within one 10,000th of a second. Sybase IQ automatically handles the conversion. Sybase IQ also has a separate `DATE` data type, a single 4-byte integer. If you want to extract just a date from a SQL Server or Adaptive Server Enterprise `datetime` column, you can do this with Sybase IQ `DATE` data type. To do this, define an Sybase IQ `DATE` column with same name as the Adaptive Server Enterprise `datetime` column. Sybase IQ automatically picks up appropriate portion of `datetime` value.
`smalldatetime`	`TIMESTAMP` or `DATETIME` or `DATE` or `TIME`	Define Adaptive Server Enterprise `smalldatetime` columns as `TIMESTAMP` (or `DATETIME`) data type in Sybase IQ. Sybase IQ properly handles the conversion. As with regular `datetime`, if you want to extract just a date from an Adaptive Server Enterprise `smalldatetime` column, do it with the Sybase IQ `DATE` data type.
`date`	`date`	You can insert data from an Adaptive Server Enterprise database column of data type `date` using **INSERT...LOCATION**.

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`time`	`time`	The Sybase IQ data type is the Time of day, containing hour, minute, second, and fraction of a second. The fraction is stored to 6 decimal places. A `time` value requires 8 bytes of storage. The Adaptive Server Enterprise data type `time` is between 00:00:00:000 and 23:59:59:999. You can use either military time or 12AM for noon and 12PM for midnight. A `time` value must contain either a colon or the AM or PM signifier. AM or PM may be in either uppercase or lowercase. A `time` value requires 4 bytes of storage. You can insert data from an Adaptive Server Enterprise database column of data type `time` using **INSERT...LOCATION**.

The following table indicates the Adaptive Server Enterprise `binary` data types and the Sybase IQ equivalents.

Table 38. Binary data types

Adaptive Server Enterprise data type	Sybase IQ data type	Notes
`binary`	`BINARY`	Sybase IQ pads trailing zeros on all `BINARY` columns. Always create **BINARY** columns with an even number of characters for length. `HNG` indexes do not allow `BINARY` data.
`varbinary`	`VARBINARY`	Sybase IQ does not pad or truncate trailing zeros on `VARBINARY` columns. Always create **VARBINARY** columns with an even number of characters for length. `HNG` indexes do not allow `VARBINARY` data.

Since the following Adaptive Server Enterprise data types are not supported, you must omit columns with these data types:

- `nchar`, `nvarchar`
- `univar`, `univarchar`
- `unsigned smallint`
- `native Java data types`

This also applies to any custom Adaptive Server Enterprise data type.

Conversion Errors on Data Import

When you load data from external sources, there may be errors in the data.

For example, there may be dates that are not valid dates and numbers that are not valid numbers. The **CONVERSION_ERROR** database option allows you to ignore conversion errors by converting them to NULL values.

For information on setting **dbisql** database options, see the **SET OPTION** statement in *Reference: Statements and Options*.

Tune Bulk Loading of Data

Loading large volumes of data into a database can take a long time and use a lot of disk space. There are a few things you can do to save time.

See also

- *Data Loads* on page 15
- *Tools for Investigating Lock Contention* on page 380
- *Schedule Database Updates* on page 243

Load Performance During Database Definition

The way you define your database, tables, and indexes can have a dramatic impact on load performance.

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488

Number of Distinct Values

Sybase IQ optimizes loading of data for a large or small set of distinct values, based on the setting of the `MINIMIZE_STORAGE` database option, and parameters you specify when you create your database and tables

. Parameters that affect load optimization include:

- The **UNIQUE** and **IQ UNIQUE** options, and the data type and width of the column, all specified in the **CREATE TABLE** or **ALTER TABLE** command.
- The **IQ PAGE SIZE**, specified in the **CREATE DATABASE** command.

See also

- *Guidelines for Creating Tables* on page 179
- *How Locking Works* on page 376
- *IQ PAGE SIZE Parameter Guidelines* on page 154
- *Rules and Checks for Valid Data* on page 341

Index Creation

To make the best use of system resources, create all of the indexes you need before loading data. While you can always add new indexes later, it is much faster to load all indexes at once.

Memory Adjustments

On some platforms you can set command-line options to adjust the amount of memory available.

Increasing memory can improve load performance.

Load Time Environment Adjustments

When you load data, you can adjust several factors to improve load performance.

- Use the **LOAD TABLE** command whenever you have access to raw data in ASCII or binary format. especially for all loads of over a hundred rows. The **LOAD TABLE** command is the fastest insertion method.
- When loading from a flat file, use binary data if you have a choice of using binary or character data. This can improve performance by eliminating conversion costs and reducing I/O.
- Set **LOAD TABLE** command options appropriately.

  Sybase strongly recommends setting the **LOAD TABLE IGNORE CONSTRAINT** option limit to a non-zero value, if you are logging the ignored integrity constraint violations. Logging an excessive number of violations affects the performance of the load.
- Place data files on a separate physical disk drive from the database file, to avoid excessive disk head movement during the load.
- Increase the size of the database cache. Providing enough memory for the load is a key performance factor. Use the database startup utility **start_iq** command line options **iqmc** and **iqtc** to increase the cache size; see the *Utility Guide > start_iq Database Server Startup Utility*. For these options to take effect, you must restart the server.
- Adjust the degree of buffer partitioning for your database or server, to avoid lock contention. Buffer partitioning based on the number of CPUs is enabled by default, and can be adjusted by setting the **-iqpartition** server command line option or the **Cache_Partitions** database option.

- Ensure that only one user at a time updates the database. While users can insert data into different tables at the same time, concurrent updates can slow performance.
- Schedule major updates for low usage times. Although many users can query a table while it is being updated, query users require CPU cycles, disk space, and memory. You will want these resources available to make your inserts go faster.
- If you are using the **INSERT** statement, run Interactive SQL or the client application on the same machine as the server if possible. Loading data over the network adds extra communication overhead. This might mean loading new data during off hours.

  If you are using **INSERT...LOCATION** to load large amounts of text or bulk data across a network from a remote Adaptive Server Enterprise database, use the PACKETSIZE parameter of the **LOCATION** clause to increase the TDS packet size. This change may significantly improve load performance.

See also

- *Bulk Loads with the LOAD TABLE Statement* on page 254
- *Tools for Investigating Lock Contention* on page 380

IQ Main Store Space Use in Incremental Loads

An incremental load may modify a large number of pages within the table being loaded.

As a result, the pages are temporarily versioned within the main dbspace, until the transaction commits and a checkpoint can release the old versions. This versioning can be particularly prevalent if the incremental load follows a delete from the same table. The reason for this is that, by default, Sybase IQ reuses row IDs from deleted records.

To help reduce space usage from versioned pages, set the **APPEND_LOAD** option ON so that IQ appends new data to the end of the table. **APPEND_LOAD** is OFF by default.

The `APPEND_LOAD` option applies to **LOAD**, **INSERT...SELECT**, and **INSERT...VALUES** statements.

Thread Use During Loads

When possible, Sybase IQ uses multithreading to improve load performance.

Fixed-width loads and full-width, row-delimited loads (where size and limit=0) will run fully multithreaded provided enough resources—memory and/or threads—are available. If there are not enough resources, the load runs on a single thread, and this message appears in the `.iqmsg` file:

```
The insert to the table will be single threaded.
```

Variable-width loads without row-delimited data will run only partially multithreaded at best, provided there are enough resources to do so. For loads that are partially multithreaded, the following message appears in the `.iqmsg` file:

```
Portions of the insert/load will be multithreaded.
```

Changes to Table Rows

To update one or more rows, use the **UPDATE** statement. The new data can be a constant or an expression that you specify or data pulled from other tables.

As in all data modification statements, you can change the data in only one table or view at a time.

If an **UPDATE** statement violates an integrity constraint, the update does not take place and an error message appears. For example, if one of the values being added is the wrong data type, or if it violates a constraint defined for one of the columns or data types involved, the update does not take place.

A simplified version of the syntax is:

```
UPDATE table-name
SET column_name = expression
WHERE search-condition
```

Examples

If the company Newton Ent. (in the Customers table of the demo database) is taken over by Einstein, Inc., you can update the name of the company using a statement such as the following:

```
UPDATE Customers
SET company_name = 'Einstein, Inc.'
WHERE company_name = 'Newton Ent.'
```

You can use any condition in the WHERE clause. If you are not sure how the company name was entered, you could try updating any company called Newton, with a statement such as the following:

```
UPDATE Customers
SET company_name = 'Einstein, Inc.'
WHERE company_name LIKE 'Newton%'
```

The search condition need not refer to the column being updated. The company ID for Newton Entertainments is 109. As the ID value is the primary key for the table, you could be sure of updating the correct row using the following statement:

```
UPDATE Customers
SET company_name = 'Einstein, Inc.'
WHERE id = 109
```

The SET Clause

The **SET** clause specifies the columns to be updated, and their new values. The **WHERE** clause determines the row or rows to be updated. If you do not have a **WHERE** clause, the specified columns of all rows are updated with the values given in the **SET** clause.

You can provide any expression of the correct data type in the SET clause.

The WHERE Clause

The **WHERE** clause specifies the rows to be updated. For example, the following statement replaces the One Size Fits All Tee Shirt with an Extra Large Tee Shirt

```
UPDATE Products
SET size  = 'Extra Large'
WHERE name = 'Tee Shirt'
   AND size = 'One Size Fits All'
```

The FROM Clause

You can use a **FROM** clause to pull data from one or more tables into the table you are updating. You can also employ a **FROM** clause to use selection criteria against another table to control which rows are updated.

Ways to Delete Data

Use the **DELETE**, **DROP TABLE**, and **TRUNCATE TABLE** statements to delete data.

To remove data from a database, you can do any of the following:

- Use the **DELETE** statement to remove from a table all rows that meet the criteria you specify.
- Use the **DROP TABLE** statement to remove an entire table, including all data rows.
- Use the **TRUNCATE TABLE** statement to delete all rows from a table, without deleting the table definition.

Space for Deletions

When you use the **DELETE** or **TRUNCATE TABLE** statement, you may need to add space to your database, due to the way Sybase IQ stores versions of data pages.

When you use **DROP TABLE**, you do not need to add space, as no extra version pages are needed.

Managing User IDs and Permissions

Each user of a database must be assigned a unique user ID: the name to type when connecting to the database. This chapter describes how to manage user IDs.

See also

- *Permissions for Modifying Data* on page 243
- *Security Overview* on page 4
- *Transactions and Versioning* on page 365

Database Permissions and Authorities Overview

Proper management of user IDs and permissions allows you to work effectively while maintaining the security and privacy of appropriate information within the database.

Use SQL statements to assign user IDs to new users of a database, to grant and revoke permissions and authorities for database users, and to display the current permissions of users. You can also use Sybase Control Center to assign user IDs and to grant and revoke authorities; for instructions, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

A *permission* grants the ability to create, modify, query, use, or delete database objects such as tables, views, users, and so on. An *authority* grants the ability to perform a task at the database level, such as backing up the database.

Database permissions are assigned to user IDs. Throughout this chapter, the term *user* serves as a synonym for user ID. Remember, however, that permissions are granted and revoked for each user ID.

Setting Up Individual User IDs

Even if there are no security concerns regarding a multiuser database, there are good reasons for setting up an individual user ID for each user. The administrative overhead for individual user IDs is very low if a group with the appropriate permissions is set up. Groups of users are discussed later in this chapter.

Among the reasons for using individual user IDs are the following:

- The network server screen and the listing of connections in Sybase Central are both much more useful with individual user IDs, as you can tell which connections are which users.
- The backup log identifies the user ID that created the backup.
- The message log displays the user ID for each database connection.

While all permissions are inheritable (from the groups to which the user belongs), only some authorities are inheritable.

Except for DBA, which has full administrative privileges, each authority has permissions to perform certain types of tasks.

See also

- *Learning Roadmap for Connections* on page 46

Inheriting Authorities

Some authorities are inherited through group membership.

Table 39. Authorities

Authority	Inherited through group membership
BACKUP	No
DBA	No
MULTIPLEX ADMIN	Yes
OPERATOR	Yes
PERMS ADMIN	Yes
PROFILE	Yes
READCLIENTFILE	Yes
READFILE	Yes
REMOTE DBA	No
RESOURCE	No
SPACE ADMIN	Yes
USER ADMIN	Yes
VALIDATE	No
WRITECLIENTFILE	Yes

For MULTIPLEX ADMIN authority, see *Using Sybase IQ Multiplex*.

Some of the authorities are provided by SQL Anywhere. See *SQL Anywhere 11.0.1 > SQL Anywhere Server - Database Administration > Configuring Your Database > Managing user IDs, authorities, and permissions > Database permissions and authorities overview > Authorities overview*.

Granting or revoking the authorities MULTIPLEX ADMIN, OPERATOR, PERMS ADMIN, SPACE ADMIN, USER ADMIN on a database created in versions earlier than Sybase IQ 15.2 returns the error code -1347:

```
Use of authority '%1' is not supported on this current
database.
```

See also

- *Authorities Overview* on page 305

Inherited Permissions

This table lists the permissions that you can assign to users, and whether they are inherited through group membership.

Permission	Inherited through group membership
ALL	Yes
ALTER	Yes
CONNECT	No
DELETE	Yes
INSERT	Yes
INTEGRATED LOGIN	No
KERBEROS LOGIN	No
REFERENCES	Yes
SELECT	Yes
UPDATE	Yes

Authorities Overview

The following sections provide details about authorities.

See also

- *Inheriting Authorities* on page 304
- *How User Permissions are Assessed* on page 335

DBA Authority Overview

When you create a database, a single user ID is created.

This first user ID is `DBA` and the password is initially set to `sql` by default. To override the default user name or password, use the **CREATE DATABASE** statement with the **DBA USER** or **DBA PASSWORD** clause. The DBA user ID is automatically given DBA permissions, also called DBA authority, within the database. This level of permission enables the `DBA` user ID to carry out any activity in the database: create tables, change table structures, create new user IDs, revoke permissions from users, and so on.

Note: To ensure database security, the DBA must change the password from the default of **sql** to a new value.

Users with DBA Authority

A user with DBA authority is referred to as the *database administrator* or *database owner*. In this section, frequent reference is made to the database administrator, or the *DBA*. This is shorthand for any user or users with DBA authority.

Although DBA authority may be granted or transferred to other user IDs, in this chapter it is assumed that the DBA user ID is the database administrator, and the abbreviation DBA is used interchangeably to mean both the DBA user ID and any user ID with DBA authority.

Warning! Never drop the DBA user for a multiplex database. Doing so makes the database invalid.

Example

The following example shows how to give non-DBA users the ability to execute commands that require DBA privileges. This example creates a policy that lets a non-DBA user (`user1`) perform backup.

```
CREATE PROCEDURE "DBA".do_backup()
BEGIN
   BACKUP DATABASE
      CRC ON
      ATTENDED OFF
      BLOCK FACTOR 4
      FULL
      TO 'fileA' SIZE 2000
      TO 'fileB' SIZE 2000
      TO 'fileC' SIZE 2000;
END;
GRANT EXECUTE ON "DBA".do_backup TO user1;
```

Adding New Users

The DBA has the authority to add new users to the database. As users are added, they are also granted permissions to carry out tasks on the database. Some users may need to simply look at the database information using SQL queries, others may need to add information to the database, and others may need to modify the structure of the database itself. Although some of the responsibilities of the DBA may be handed over to other user IDs, the DBA is responsible for the overall management of the database by virtue of the DBA authority.

The DBA has authority to create database objects and assign ownership of these objects to other user IDs.

DBA User ID in Case Sensitive Databases

User IDs and passwords are actually objects in the database. For details about case-sensitivity of user IDs and passwords, see *Reference: Building Blocks, Tables, and Procedures*.

OPERATOR Authority Overview

OPERATOR authority has the permission to checkpoint databases, drop connections (including those for users with DBA authority), back up databases, and monitor the system.

OPERATOR authority may be granted only by a user with DBA or PERMS ADMIN authority to other users. Users without DBA authority must have OPERATOR authority to perform **BACKUP**, **CHECKPOINT**, or **DROP CONNECTION** commands.

PERMS ADMIN Authority Overview

PERMS ADMIN authority is the permission to manage data permissions, groups, authorities, and passwords.

PERMS ADMIN or DBA authority is required to:

- Grant or revoke the SELECT, INSERT, UPDATE, DELETE, ALTER, or REFERENCES permissions on any applicable user object (such as a table, view, or column).
- Grant or revoke EXECUTE permission on any applicable user object (such as a procedure or function).
- Create and drop groups and grant or revoke group membership.
- Grant or revoke all existing and new authorities except DBA and REMOTE DBA.
  Change the password of any account.

Only users with DBA or PERMS ADMIN authority may grant PERMS ADMIN authority to other users.

RESOURCE Authority Overview

RESOURCE authority is the permission to create database objects, such as tables, views, and stored procedures. Only a user with DBA or PERMS ADMIN authority may grant RESOURCE authority to other users.

SPACE ADMIN Authority Overview

SPACE ADMIN authority is the permission to manage dbspaces.

SPACE ADMIN authority is required to:

- Create, alter, drop or comment on dbspaces.
- Grant or revoke CREATE permission on dbspaces.
- Move data and metadata to dbspaces.
- Perform read-only selective restore operations.

Note: SPACE ADMIN authority can create a dbspace; however to create objects in the dbspace, a user must be granted CREATE permission on the dbspace. A user who has both SPACE ADMIN and RESOURCE authority must still be granted CREATE privilege on that dbspace to create objects in that dbspace. Without CREATE privilege, attempts to create objects return `Permission Denied` errors. To add CREATE privilege, use **GRANT CREATE ON**.

Only a user with DBA or PERMS ADMIN authority may grant SPACE ADMIN authority to other users.

See also

- *Before You Restore* on page 427

USER ADMIN Authority Overview

USER ADMIN authority is the permission to manage users, external logins, and login policies.

USER ADMIN authority is required to:

- Create and drop database users, and assign initial passwords.
- Create, alter, or drop login policies, and assign login policies to users.
- Define user authentication mechanism, such as Kerberos or integrated login.
- Create or drop external logins.
- Force password change on next login for users.
- Reset user login policies.

Only a user with DBA or PERMS ADMIN authority may grant USER ADMIN authority to other users.

Ownership Permissions Overview

Ownership of a database object carries with it permissions to carry out actions on that object. The creator of a database object becomes the owner of that object.

Ownership permissions are not assigned to users in the same way that other permissions in this chapter are assigned.

Owners

A user who creates a new object within the database is called the *owner* of that object, and automatically has permission to carry out any operation on that object. The owner of a table may modify the structure of that table, for instance, or may grant permissions to other database users to update the information within the table.

Note: The owner of a table can only load that table if he or she is DBA or the server was started with the **-gl all** switch on the command line or configuration file. Ownership and resource authority are not sufficient to use **LOAD TABLE**. In order to use the **LOAD TABLE** statement, you also need INSERT permission on the table.

The DBA has permission to modify any component within the database, and so could delete a table created by another user, for instance. The DBA has all the permissions regarding database objects that the owner of each object has.

The DBA is also able to create database objects for other users, and in this case the owner of an object is not the user ID that executed the CREATE statement. Despite this possibility, this chapter refers interchangeably to the owner and creator of database objects.

See also

- *Groups Without Passwords* on page 328

Dbspace Management Permissions

To create a table object or join index in a dbspace or to move a table object or join index to a new dbspace requires CREATE permission in the dbspace.

The CREATE permission in a dbspace can be granted/revoked to/from a user or a group. Any member in a group inherits CREATE permission from the group. By default, CREATE permission on IQ_SYSTEM_MAIN, IQ_SYSTEM_TEMP, and SYSTEM is granted to PUBLIC. For other IQ main dbspaces, the system administrator must explicitly grant CREATE permission on the dbspace to a group/user before they can create or move objects into that dbspace. For example, if a table is to be placed on a new IQ main dbspace, the user must have CREATE permission on that dbspace. Users must also have RESOURCE permission to create objects.

See also

- *Disconnecting from a Database in Embedded SQL* on page 160

Table and Views Permissions Overview

There are several distinct permissions that may be granted to user IDs concerning tables and views:

Permission	Description
ALTER	Permission to alter the structure of a table
DELETE	Permission to delete rows from a table or view
INSERT	Permission to insert rows into a table or view
REFERENCES	Permission to create indexes on a table, and to create foreign keys that reference a table
SELECT	Permission to look at information in a table or view
UPDATE	Permission to update rows in a table or view. This may be granted on a set of columns in a table only
ALL	All the above permissions

In a multiplex, only write servers can modify table permissions on tables owned by the write server.

Group Permissions Overview

Setting permissions individually for each user of a database can be a time-consuming and error-prone process. For most databases, permission management based on groups, rather than on individual user IDs, is a much more efficient approach.

You can assign permissions to a group in exactly the same way as to an individual user. You can then assign membership in appropriate groups to each new user of the database, and they gain a set of permissions by virtue of their group membership.

Example

For example, you may create groups for different departments in a company database (`sales`, `marketing`, and so on) and assign these groups permissions. Each salesperson is made a member of the `sales` group, and automatically gains access to the appropriate areas of the database.

Any user ID can be a member of several groups, and inherits all permissions from each of the groups.

Server Command-Line Permission Options

The database server startup command **start_iq** has options that set the permission level of some database and server functions.

Table 40. Startup options affecting permissions

Option	Description	Allowed values	Default
-gd *level*	Set permission required to start the database	DBA, ALL, NONE	DBA
-gk *level*	Set permission required to stop the server	DBA, ALL, NONE	DBA
-gl *level*	Set permission required to load data	DBA, ALL, NONE	ALL for servers started with **start_iq**; DBA for other servers
-gu *level*	Set permission required to execute utility commands, for example **CREATE DATABASE** and **DROP DATABASE**	DBA, ALL, NONE, UTILITY_DB	ALL

See *System Administration Guide: Volume 1 > Sybase IQ Startup Command Line Options that Control Permissions* and *Utility Guide > start_iq Database Server Startup Utility* for more details on these options and the permission level values and defaults.

Login Management

Sybase IQ defines the rules to be followed when establishing a user's database connection in a database object called a *login policy*.

A login policy is a named object in the database that consists of a set of options. Each login policy is associated with a set of options called login policy options.

You must have DBA privileges or USER ADMIN authority to create new login policies or assign an existing login policy to a user. For the SQL command syntax to manage policies, see ALTER LOGIN POLICY statement, ALTER USER statement, CREATE LOGIN POLICY statement, and DROP LOGIN POLICY statement in *Reference: Statements and Options*.

Each new database is created with a default login policy, called the root policy. You can modify the option values for the root login policy, but you cannot drop the policy. When you create a user account without specifying a login policy, the user becomes part of the root login policy. Any options that are not explicitly set when creating a login policy inherit their values from the root policy.

See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > Managing user IDs, authorities, and permissions*.

You can execute login management commands on any multiplex server; they automatically propagate to all servers in the multiplex. For best performance, Sybase recommends that you execute these commands, or any DDL, on the coordinator. See *Using Sybase IQ Multiplex*.

For instructions on managing login policies with Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

Migrating databases to Sybase IQ 15.3 from 12.7 removes existing login management settings. To re-create login management settings after migration, use the SQL syntax in the following section.

User Account and Connection Management

The Sybase IQ login management facility helps you manage users and connections to a database.

Users with DBA or USER ADMIN authority can add or drop users and control connections by:

- Limiting the number of active logins for a single user.

  To do this, assign a user to a login policy in which you have specified the **max_connections** login policy option.

- Locking out a user.

  To do this, assign a user to a login policy that has the **locked** option set ON.
- Setting user password expirations.

  To do this, specify the **max_failed_login_attempts** and **max_days_since_login** for a login policy and assign a user to this policy. You can also explicitly expire a user password by using the **FORCE PASSWORD CHANGE** clause in the SQL statement **CREATE/ALTER USER**.

For the SQL command syntax to manage policies, see CREATE LOGIN POLICY, ALTER LOGIN POLICY, and DROP LOGIN POLICY in Reference: Statements and Options.

Each new database contains a login policy named `root`. You can modify the option values for the `root` login policy, but you cannot drop the policy. When a user account is created without specifying its login policy, the user becomes part of the `root` login policy.

Stored Procedures for Login Management

This table lists the procedure you call to perform each Sybase IQ login management function.

Table 41. Stored procedures for login management

Stored procedure	Purpose	Authority required
sa_get_user_status	Retrieve the current status of all existing users	DBA or USER ADMIN
sp_expireallpasswords	Cause all user passwords to expire immediately	DBA or USER ADMIN
sp_iqaddlogin	Add users, define their passwords, specify login policy and password expiry on next login	DBA
sp_iqcopyloginpolicy	Create a new login policy by copying an existing one	DBA
sp_iqdroplogin	Drop the specified user	DBA
sp_iqmodify	Assign a given user to a login policy	DBA
sp_iqmodifyadmin	Set an option on a named login policy to a certain value	DBA
sp_iqpassword	Change a user's password	DBA to change another user's password. All users can run sp_iqpassword to change their own passwords.

See also

- *Managing Individual User IDs and Permissions* on page 316
- *Limits on Database Connections* on page 337

Preventing Connection After Failed Login Attempts

This example shows how you can prevent a user from connecting after five failed login attempts.

1. Create a login policy `lp` that has its login policy option **max_failed_login_attempts** set to the value 5.

   ```
   CREATE LOGIN POLICY lp max_failed_login_attempts=5;
   ```

2. Create a user `John` who belongs to the login policy `lp`.

   ```
   CREATE USER john IDENTIFIED BY j345 LOGIN POLICY lp;
   ```

   Because John belongs to `lp` login policy where **max_failed_login_attempts**=5, then this user will not be able to log in to the IQ server as soon as he exceeds the limit 5 for **max_failed_login_attempts**.

Locking Out Users

You can force a user to be locked out by having a login policy specifically for locked-out users.

Create a login policy with login policy option **locked**=ON, as follows:

1.

   ```
   CREATE LOGIN POLICY locked_users locked=ON
   ```

2. Assign a user whom you want to lock out to the `locked_users` policy, for example:

   ```
   ALTER USER john LOGIN POLICY locked_users
   ```

Unlocking Users

There are several ways to unlock users.

1. Assume that a user is locked because he or she belongs to a login policy with **locked**=ON. To unlock such a user, assign him or her to a login policy where option **locked**=OFF.
2. A user can also be locked if he/she has exceeded the **max_failed_login_attempts** or **max_days_since_login**. To unlock such a user, use a statement like the following:

   ```
   ALTER USER john RESET LOGIN POLICY
   ```

   The preceding statement is semantically equivalent to `ALTER USER userid LOGIN POLICY current-policy-for-user`.

Utility Database Server Security

Sybase IQ includes a phantom database, called the *utility database*, that has no physical representation.

There is no database file for this database and the database can contain no data. The utility database can run on any Sybase IQ server. In Sybase Central, the server for the utility database is known as the Utility Server.

The utility database permits a narrow range of specialized functions. It is provided so that you can execute database file manipulation statements such as **CREATE DATABASE** and **DROP DATABASE** without first connecting to a physical database.

You can also retrieve database and connection properties from the utility database. These properties apply to databases you create when connected to the utility database.

One of your configuration tasks is to set up security for the utility database and its server. There are two aspects to utility database server security:

- Who can connect to the utility database?
- Who can execute file administration statements?

See also

- *Before You Restore* on page 427
- *Command Line Options that Control Permissions* on page 30
- *Moving Database Files* on page 431
- *The Utility Database* on page 14
- *Verifying a Database Backup* on page 439
- *CREATE DATABASE Statement Defaults* on page 150
- *Database Creation with SQL* on page 147
- *Multiplex Capability* on page 5
- *Relative Path Names* on page 151

Starting the Utility Database

You cannot specify a database file when starting the utility database, because no database file is associated with that database.
Specify `utility_db` as the database name when connecting.

For example:

```
                              dbisqlc -c
"uid=dba;pwd=sql;eng=myserver;dbn=utility_db"
```

Note: When you connect to the utility database to create an IQ database having Windows raw partitions, note that there is a syntax difference in the IQ PATH. For example, to specify a

Windows raw partition on device I: for the utility database, you can use the specification "\\.\I:" On other IQ databases, you must double the slash characters, so that the same device would be specified "\\\\.\\I:". The backslash character is treated as an escape character in IQ databases but as a normal character in the utility database.

Defining the Utility Database Password

To use the utility database, you must specify the user ID `DBA`.

1. With a text editor, open the file `util_db.ini`, which is stored in the server executable directory.

   Because this directory is on the server, you can control access to the file, and thereby control who has access to the password.
2. Edit the following line, replacing the word "password" with the desired password:

   ```
   [UTILITY_DB]
   PWD=password
   ```

   Use of the utility_db security level relies on the physical security of the computer hosting the database server, since the `util_db.ini` file can be easily read using a text editor.

Permission to Execute File Administration Statements

To provide additional database security, a separate level of security controls creating and dropping databases. The **-gu** database server command-line option controls who can execute the file administration statements.

There are four levels of permission for the use of file administration statements. These levels are: `all`, `none`, `DBA`, and `utility_db`. The `utility_db` level permits only a person able to connect to the utility database to use the file administration statements.

Table 42. Permissions for file administration

-gu switch value	Effect	Applies to
all	Anyone can execute file administration statements	Any database including the utility database
none	No one can execute file administration statements	Any database including the utility database
DBA	Only DBA-authority users can execute file administration statements	Any database including the utility database
utility_db	Only the users who can connect to the utility database can execute file administration statements	Only the utility database

Examples

On Sun, HP, Linux, and Windows platforms, to permit only the user knowing the utility database password to connect to the utility database and create or delete databases, start the server at the command line with the following command:

```
start_iq -n testsrv -gu utility_db
```

On AIX, to permit only the user knowing the utility database password to connect to the utility database and create or delete databases, start the server at the command line with the following command:

```
start_iq -n testsrv -gu utility_db -iqmt 256
```

Assuming that the utility database password has been set during installation to `IQ&Mine49`, the following command starts the Interactive SQL utility as a client application, connects to the server named `testsrv`, loads the utility database and connects the user:

```
              dbisql -c
"uid=DBA;pwd=IQ&Mine49;dbn=utility_db;eng=testsrv"
```

Executing this statement successfully connects you to the utility database. You are now able to create and delete databases.

Note: The database name, user ID, and password are case sensitive. Make sure that you specify the same case in the **dbisql** command and the `util_db.ini` file.

Managing Individual User IDs and Permissions

Before you manage groups, you should understand user permissions. You can grant permissions to users and create new users using Interactive SQL and Sybase Central.

For most databases, the bulk of permission management should be carried out using groups, rather than by assigning permissions to individual users one at a time. However, because groups are simply a user ID with special properties attached, you should read and understand this section before moving on to the discussion of managing groups.

Using IQ Stored Procedures to Manage Users

You can also create new users using IQ system procedures. You must use those procedures to add users and modify their passwords and other login capabilities, in order to manage those users with the Sybase IQ Login Management facility.

To add and modify users with Sybase IQ Login Management, use the system procedures for user account and connection management.

To grant users permissions on database objects, you must still use the commands and procedures described in the rest of this section.

Using ASE Stored Procedures to Manage Users

This chapter explains how to manage users and groups using Interactive SQL and Sybase Central. You can perform many of the same tasks using Adaptive Server Enterprise-compatible stored procedures. If you have previously used Adaptive Server Enterprise or pre-Version 12.0 Sybase IQ, you may prefer to use these stored procedures. See *Reference: Building Blocks, Tables, and Procedures*. The ASE stored procedures do not let you use the Sybase IQ Login Management facilities for limiting connections.

See also

- *User Account and Connection Management* on page 311
- *Group Management* on page 325

Creating a User with Interactive SQL

The DBA or user with USER ADMIN authority can create new users using the **CREATE USER** statement.

1. From Interactive SQL, connect to the database as a user with DBA or USER ADMIN authority.
2. Issue the **CREATE USER** statement.

   This example adds user ID `M_Haneef` to a database with password *welcome*.

   ```
   CREATE USER M_Haneef
   IDENTIFIED BY welcome
   ```

Creating a User in Sybase Central

Follow these steps to create users.

1. Connect to the database.
2. Right-click the Users & Groups folder.
3. Choose New > User. The Create User wizard leads you through the process.

For more information, see the Sybase Central Online Help for the IQ plug-in.

Creating a User in Sybase Control Center

For instructions on creating a Sybase IQ user in Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

SCC provides tools for managing users and groups both for Sybase IQ and for SCC itself. Be sure to find the instructions for Sybase IQ users and groups.

Changing a Password

If you have DBA or PERMS ADMIN authority, you can change the password of any existing user.

Use this command to change a password:

```
ALTER USER userid IDENTIFIED BY password
```

If you inadvertently enter the user ID of an existing user when you mean to add a new user, you are actually changing the password of the existing user. You do not receive a warning because this behavior is considered normal. This behavior differs from pre-Version 12 Sybase IQ.

To avoid this situation, use the `sp_iqaddlogin` system procedure or the **CREATE USER** to add users. You may also use the `sp_addlogin` and `sp_adduser` system procedures. These procedures give you an error if you try to add an existing user ID, as in Adaptive Server Enterprise and pre-Version 12 Sybase IQ.

Password Rules

You can set up password rules and verify that any new password assigned complies with them. For example, you might require that passwords must include one digit or must not be the user ID. For details, see the **VERIFY_PASSWORD_FUNCTION** option in *Reference: Statements and Options*.

To set a minimum password length, see **MIN_PASSWORD_LENGTH** option in *Reference: Statements and Options*.

Changing the DBA Password

The default password for user ID DBA for all databases is `sql`. You should change this password to prevent unauthorized access to your database.

Prerequisites

To change the DBA password, you must have DBA or PERMS ADMIN authority.

If you are using **dbisql**, it is a good idea to put your permission grants into a command file for reference and so that it can be modified and run again if it is necessary to recreate the permissions.

Task

The user ID DBA identifies a user with full administration and resource creation rights. The following command changes the password for user ID DBA to new_password:

```
ALTER USER DBA
IDENTIFIED BY new_password
```

Granting DBA Authority to a User

DBA authority can grant any authority. Only the DBA can grant DBA authority to database users.

DBA authority is very powerful, granting the ability to carry out any action on the database and access to all the information in the database. It is generally inadvisable to grant DBA authority to more than a very few people.

1. Connect to the database as a user with DBA authority.
2. Type and execute the following SQL statement:

```
GRANT DBA TO userid
```

You should give two user IDs to users with DBA authority—one with DBA authority, and one without—so that they connect as DBA only when necessary.

Granting OPERATOR Authority to a User

Most authorities are granted in the same manner. DBA authority can grant any authority. PERMS ADMIN authority can grant any authority except DBA and REMOTE DBA.

1. Connect to the database as a user with DBA or PERMS ADMIN authority.
2. Type and execute the following SQL statement:

```
GRANT OPERATOR TO userid
```

Granting PERMS ADMIN Authority to a User

PERMS ADMIN authority can grant any authority except DBA and REMOTE DBA.

1. Connect to the database as a user with DBA or PERMS ADMIN authority.
2. Type and execute the following SQL statement:

```
GRANT PERMS ADMIN TO userid
```

Granting RESOURCE Authority to a User

RESOURCE authority allows the user to create new database objects, such as tables, views, indexes, or procedures.

Most authorities are granted in the same manner. DBA authority can grant any authority. PERMS ADMIN authority can grant any authority except DBA and REMOTE DBA.

1. Connect to the database as a user with DBA or PERMS ADMIN authority.
2. Type and execute the following SQL statement:

```
GRANT RESOURCE TO userid
```

Granting SPACE ADMIN Authority

Most authorities are granted in the same manner. DBA authority can grant any authority. PERMS ADMIN authority can grant any authority except DBA and REMOTE DBA.

1. Connect to the database as a user with DBA or PERMS ADMIN authority.
2. Type and execute the following SQL statement:

```
GRANT SPACE ADMIN TO userid
```

Granting USER ADMIN Authority to a User

USER ADMIN authority lets you manage external logins, login policies, and other users.

1. Connect to the database as a user with DBA or PERMS ADMIN authority.
2. Type and execute the following SQL statement:

```
GRANT USER ADMIN TO userid
```

Permissions on Tables and Views

You can assign a set of permissions on individual tables and views. Users can be granted combinations of these permissions to define their access to a table or view.

Combinations of Permissions

- The ALTER (permission to alter the structure of a table) and REFERENCES (permission to create indexes and to create foreign keys) permissions grant the authority to modify the database schema, and so will not be assigned to most users. These permissions do not apply to views.
- The DELETE, INSERT, and UPDATE permissions grant the authority to modify the data in a table or view. Of these, the UPDATE permission may be restricted to a set of columns in the table or view.
- The SELECT permission grants authority to look at data in a table or view, but does not give permission to alter it.
- ALL permission grants all the above permissions.

Granting Delete Permission on Tables and Views in Interactive SQL

All table and view permissions are granted in a very similar fashion.

Use the GRANT statement to grant permissions. For example, to grant permission to M_Haneef to delete rows from the table named `sample_table`:

1. Connect to the database as a user with DBA or PERMS ADMIN authority, or as the owner of `sample_table`.
2.

   Type and execute the SQL statement:

```
GRANT DELETE
ON sample_table
TO M_Haneef
```

See also

- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting Permissions on Procedures in Sybase Central* on page 323
- *Granting the Right to Grant Permissions* on page 322
- *Granting Update Permission on Tables and Views in Interactive SQL* on page 321
- *Granting User Permission on Tables in Sybase Central* on page 321
- *Permissions of Groups* on page 327

Granting Update Permission on Tables and Views in Interactive SQL

able and view permissions are limited in that they apply to all the data in a table or view (except for the UPDATE permission which may be restricted).

T Finer tuning of user permissions can be accomplished by creating procedures that carry out actions on tables, and then granting users the permission to execute the procedure.

To grant permission to M_Haneef to update the `column_1` and `column_2` columns only in the table named `sample_table`:

Connect to the database as a user with DBA or PERMS ADMIN authority, or as the owner of `sample_table`.

Type and execute the SQL statement:

```
GRANT UPDATE (column_1, column_2)
ON sample_table
TO M_Haneef
```

See also

- *Granting Delete Permission on Tables and Views in Interactive SQL* on page 320
- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting Permissions on Procedures in Sybase Central* on page 323
- *Granting the Right to Grant Permissions* on page 322
- *Granting User Permission on Tables in Sybase Central* on page 321
- *Permissions of Groups* on page 327

Granting User Permission on Tables in Sybase Central

One way to grant a user permissions on a table in Sybase Central is as follows:

1. Connect to the database.
2. Double-click the Tables folder for that database, to display the tables in the left panel.
3. Right-click a table and choose Properties from the popup menu.

4. On the Permissions tab of the Properties dialog, configure the permissions for the table:

   - Click Grant to select users or groups to which to grant full permissions.
   - Click in the fields beside the user or group to set specific permissions. Permissions are indicated by a check mark, and grant options are indicated by a check mark with two '+' signs.
   - Select a user and click the button beside References, Select, or Update to set that type of permission on individual columns.
   - Select a user or group in the list and click Revoke to revoke all permissions.

You can also assign permissions from the Users & Groups property sheet. To assign permissions to many users and groups at once, use the table's property sheet. To assign permissions to many tables at once, use the user's property sheet.

See also

- *Granting Delete Permission on Tables and Views in Interactive SQL* on page 320
- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting Permissions on Procedures in Sybase Central* on page 323
- *Granting the Right to Grant Permissions* on page 322
- *Granting Update Permission on Tables and Views in Interactive SQL* on page 321
- *Permissions of Groups* on page 327

Granting the Right to Grant Permissions

To assign table and view permissions, use the **GRANT** statement clause **WITH GRANT OPTION**.

This option gives the right to pass on the permission to other users.

For example, to grant permission to M_Haneef to delete rows from the table named `sample_table`, and the right to pass on this permission to other users:

1. Connect to the database as a user with DBA or PERMS ADMIN authority, or as the owner of `sample_table`:
2. Type and execute the SQL statement:

```
GRANT DELETE ON sample_table
TO M_Haneef
WITH GRANT OPTION
```

See also

- *Granting Delete Permission on Tables and Views in Interactive SQL* on page 320
- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting Permissions on Procedures in Sybase Central* on page 323
- *Granting Update Permission on Tables and Views in Interactive SQL* on page 321
- *Granting User Permission on Tables in Sybase Central* on page 321

• *Permissions of Groups* on page 327

Granting Permissions on Procedures in Interactive SQL

There is only one permission that may be granted on a procedure – the EXECUTE permission to execute (or CALL) the procedure.

Permission to execute stored procedures may be granted by the DBA or PERMS ADMIN or by the owner of the procedure (the user ID that created the procedure).

Use the GRANT EXECUTE statement to grant permissions on procedures. For example, to grant M_Haneef permission to execute a procedure named `my_procedure`:

1. Connect to the database as a user with DBA or PERMS ADMIN authority or as owner of `my_procedure` procedure.
2. Execute the SQL statement:

```
GRANT EXECUTE
ON my_procedure
TO M_Haneef
```

Execution Permissions of Procedures

Procedures execute with the permissions of their owner. Any procedure that updates information on a table will execute successfully only if the owner of the procedure has UPDATE permissions on the table.

As long as the procedure owner does have the proper permissions, the procedure will execute successfully when called by any user assigned permission to execute it, whether or not they have permissions on the underlying table. You can use procedures to allow users to carry out well-defined activities on a table, without having any general permissions on the table.

See also

- *Granting Delete Permission on Tables and Views in Interactive SQL* on page 320
- *Granting Permissions on Procedures in Sybase Central* on page 323
- *Granting the Right to Grant Permissions* on page 322
- *Granting Update Permission on Tables and Views in Interactive SQL* on page 321
- *Granting User Permission on Tables in Sybase Central* on page 321
- *Permissions of Groups* on page 327
- *Procedures Provide Tailored Security* on page 334

Granting Permissions on Procedures in Sybase Central

One way to grant a user permissions on a table in Sybase Central is as follows.

1. Connect to the database.
2. Click the **Users & Groups** folder, and locate the user you want to grant permissions to.

3. Right-click the user, and select **Copy** from the popup menu.
4. Locate the procedure you want to allow the user to execute, in the Stored Procedures folder.
5. Click the procedure, and choose **Edit > Paste** from the main menu to grant permissions.

For more information, see the Sybase Central online Help.

See also

- *Granting Delete Permission on Tables and Views in Interactive SQL* on page 320
- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting the Right to Grant Permissions* on page 322
- *Granting Update Permission on Tables and Views in Interactive SQL* on page 321
- *Granting User Permission on Tables in Sybase Central* on page 321
- *Permissions of Groups* on page 327
- *Procedures Provide Tailored Security* on page 334

Revoking User Permissions in Interactive SQL

Any user's permissions are a combination of those that have been granted and those that have been revoked. By revoking and granting permissions, you can manage the pattern of user permissions on a database.

The REVOKE statement is the exact converse of the GRANT statement.

1. Connect as a user with DBA or PERMS ADMIN authority or as the owner of the procedure.
2. Execute a REVOKE statement.

   For example, to disallow M_Haneef from executing `my_procedure`:

   ```
   REVOKE EXECUTE ON my_procedure FROM M_Haneef
   ```

   To disallow M_Haneef from deleting rows from `sample_table`:

   ```
   REVOKE DELETE ON sample_table FROM M_Haneef
   ```

Warning! Before you revoke privileges or drop a user, be aware of the following restrictions:

- Before issuing **REVOKE CONNECT** or **sp_dropuser**, you must remove any objects, such as tables, owned by that user. If you try to revoke a user's connect privileges or use the stored procedure **sp_dropuser** while the user owns any database objects, you receive an error.
- Procedures like **sp_dropuser** provide minimal compatibility with Adaptive Server Enterprise stored procedures. If you are accustomed to Adaptive Server Enterprise (or Sybase IQ 11.x) stored procedures, you should compare their text with Sybase IQ 15.3 procedures before using the procedure in **dbisql**. To compare, use the command

  ```
  sp_helptext 'owner.procedure_name'
  ```

For all system stored procedures delivered by Sybase, the owner is `dbo`. To see the text of a stored procedure of the same name owned by a different user, you must specify that user, for example:

```
sp_helptext 'myname.myprocedure'
```

Group Management

Once you understand how to manage permissions for individual users (as described in the previous section) working with groups is straightforward.

DBA, RESOURCE, and GROUP Permissions

When permissions on tables, views, and procedures are granted to or revoked from a group, all members of the group inherit those changes. The DBA, RESOURCE, and GROUP permissions are not inherited: they must be assigned individually to each individual user ID requiring them.

A group is simply a user ID with special permissions. Granting permissions to a group and revoking permissions from a group are done in exactly the same manner as any other user.

A group can also be a member of a group. A hierarchy of groups can be constructed, each inheriting permissions from its parent group.

A user ID may be granted membership in more than one group, so the user-to-group relationship is many-to-many.

See also

• *Managing Individual User IDs and Permissions* on page 316

Creating Groups in Interactive SQL

A group is identified by a user ID, just like a single user, but this user ID is granted the permission to have *members*.

1. Connect to the database as a user with either DBA or both USER ADMIN and PERMS ADMIN authority. USER ADMIN is needed for CREATE USER, and PERMS ADMIN is needed for GRANT GROUP.
2. Create the group's user ID just as you would any other user ID, using the following SQL statement:

```
CREATE USER personnel
IDENTIFIED BY group_password
```

3. Give the personnel user ID the permission to have members, with the following SQL statement:

```
GRANT GROUP TO personnel
```

The GROUP permission, which gives the user ID the ability to have members, is not inherited by members of a group. If this were not the case, then every user ID would automatically be a group as a consequence of membership in the special PUBLIC group.

The previous example creates a groups with a user ID and password. You can also create a group without a password in order to prevent anybody from signing on using the group user ID. .

Creating Groups in Sybase Central

You can use Sybase Central to create groups.

1. Connect to the database.
2. Right-click the Users & Groups folder.
3. Choose New > Group. A wizard leads you through the process.

For more information, see the Sybase Central online Help.

Creating Groups in Sybase Control Center

For instructions on creating a Sybase IQ group in Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

SCC provides tools for managing users and groups both for Sybase IQ and for SCC itself. Be sure to find the instructions for Sybase IQ users and groups.

Adding Group Members in Interactive SQL

The GRANT statement makes a user a member of a group.

Prerequisites

Membership in a group can be granted either by the DBA or PERMS ADMIN authority.

Task

1. Connect to the database as a user with DBA or PERMS ADMIN authority.
2. Issue an appropriate GRANT statement. For example, to grant user `M_Haneef` membership in a group `personnel` :

```
GRANT MEMBERSHIP
IN GROUP personnel
TO M_Haneef
```

When users are assigned membership in a group, they inherit all the permissions on tables, views, and procedures associated with that group. If you do not want a specific user to access a particular table, view, or procedure, then do not make that user a member of a group that has permissions on that object.

You cannot revoke permissions for a specific user within a group.

Adding Group Members in Sybase Central

When users are assigned membership in a group, they inherit all the permissions on tables, views, and procedures associated with that group.

If you do not want a specific user to access a particular table, view, or procedure, then do not make that user a member of a group that has permissions on that object.

1. Connect to the database.
2. Double-click the **Users & Groups** folder for that database, to open it. Groups are displayed in the left panel, and both users and groups are displayed in the right panel.
3. In the right panel, select the users you want to add to a group, and drag them to the group.

For more information, see the Sybase Central online Help.

Adding Group Members in Sybase Control Center

For instructions on adding members to a Sybase IQ group in Sybase Control Center, see the Sybase Control Center for Sybase IQ online help in SCC or at *http://sybooks.sybase.com/nav/summary.do?prod=10680*.

SCC provides tools for managing users and groups both for Sybase IQ and for SCC itself. Be sure to find the instructions for Sybase IQ users and groups.

Permissions of Groups

Permissions may be granted to groups in exactly the same way as to any other user ID.

Permissions on tables, views, and procedures are inherited by members of the group, including other groups and their members. There are some complexities to group permissions that database administrators need to keep in mind.

The DBA, RESOURCE, and GROUP permissions are not inherited by the members of a group. Even if the `personnel` user ID is granted RESOURCE permissions, the members of `personnel` do not have RESOURCE permissions.

Ownership of database objects is associated with a single user ID and is not inherited by group members. If the user ID `personnel` creates a table, then the `personnel` user ID is the owner of that table and has the authority to make any changes to the table, as well as to grant privileges concerning the table to other users. Other user IDs who are members of `personnel` are not the owners of this table, and do not have these rights. If, however, SELECT authority is explicitly granted to the `personnel` user ID by the DBA or by the `personnel` user ID itself, all group members do have select access to the table. In other words, only granted permissions are inherited.

See also

- *Granting Delete Permission on Tables and Views in Interactive SQL* on page 320
- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting Permissions on Procedures in Sybase Central* on page 323
- *Granting the Right to Grant Permissions* on page 322
- *Granting Update Permission on Tables and Views in Interactive SQL* on page 321
- *Granting User Permission on Tables in Sybase Central* on page 321

Tables Owned by Groups

Groups are used for finding tables and procedures in the database.

For example, the query

```
SELECT * FROM SYSGROUPS
```

will always find the table SYSGROUPS, because all users belong to the PUBLIC group and PUBLIC belongs to the SYS group which owns the SYSGROUPS table. (The SYSGROUPS table contains a list of *group_name*, *member_name* pairs representing the group memberships in your database.)

If a table `Employees` is owned by the `personnel` user ID, and if `M_Haneef` is a member of the `Personnel` group, then `M_Haneef` can refer to the `Employees` table simply as `Employees` in SQL statements. Users who are not members of the `Personnel` group need to use the qualified name `Personnel.Employees`.

Groups that Own Tables

Sybase recommends that you create a group whose only purpose is to own the tables.

Do not grant any permissions to this group, but make all users members of the group. This allows everyone to access the tables without qualifying names. You can then create permission groups and grant users membership in these permission groups as warranted.

See also

- *Owner Prefixes for Object Names* on page 330

Groups Without Passwords

Users connected to a group's user ID have certain permissions. Such users would also have ownership permissions over any tables in the database created in the name of the group's user ID.

It is possible to set up a database so that all handling of groups and their database objects is done only by the DBA or by a designated user with PERMS ADMIN authority, rather than permitting other user IDs to make changes to group membership.

This is done by disallowing connection as the group's user ID when creating the group. To do this, type the GRANT CONNECT statement without a password. For example:

```
GRANT CONNECT
TO personnel
```

creates a user ID `personnel`. This user ID can be granted group permissions, and other user IDs can be granted membership in the group, inheriting any permissions that have been given to `personnel`, but nobody can connect to the database using the `personnel` user ID, because it has no valid password.

The user ID `personnel` can be an owner of database objects, even though no user can connect to the database using this user ID. The CREATE TABLE statement, CREATE PROCEDURE statement, and CREATE VIEW statement all allow the owner of the object to be specified as a user other than that executing the statement. This assignment of ownership can be carried out only by the DBA.

See also

- *Ownership Permissions Overview* on page 308

Groups Created Automatically

When a database is created, two groups are also automatically created. These are SYS and PUBLIC. Neither of these groups has passwords, so it is not possible to connect to the database as either SYS or as PUBLIC. The two groups serve important functions in the database.

The SYS Group

The SYS group is owner of the system tables and views for the database, which contain the full description of database structure, including all database objects and all user IDs.

For a description of the system tables and views, together with a description of access to the tables, see *Reference: Building Blocks, Tables, and Procedures*.

The PUBLIC Group

When a database is created, the PUBLIC group is automatically created, with CONNECT permissions to the database and SELECT permission on the system tables.

The PUBLIC group is a member of the SYS group, and has read access for some of the system tables and views, so that any user of the database can find out information about the database schema. If you wish to restrict this access, you can REVOKE PUBLIC's membership in the SYS group.

Any new user ID is automatically a member of the PUBLIC group and inherits any permissions specifically granted to that group by the DBA. You can also REVOKE membership in PUBLIC for users if you wish.

Owner Prefixes for Object Names

The name of every database object is an identifier. In certain circumstances, you need to use the owner prefix to identify tables, view and procedures.

The rules for valid identifiers are described in *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements > Identifiers* .

In queries and sample SQL statements throughout this guide, database objects from the demo database are generally referred to using their simple name. For example:

```
SELECT *
FROM Employees
```

Tables, procedures, and views all have an owner. The owner of the tables in the demo database is the user ID DBA. In some circumstances, you must prefix the object name with the owner user ID, as in the following statement.

```
SELECT *
FROM "DBA".Employees
```

The `Employees` table reference is said to be qualified. (In this case the owner name is enclosed in double quotes, as DBA is a SQL keyword.) In other circumstances it is sufficient to give the object name.

When referring to a database object, a prefix is required unless:

- You are the owner of the database object.
- The database object is owned by a group ID of which you are a member.

Example

Consider the following example of a corporate database. All the tables are created by the user ID `company`. This user ID is used by the database administrator and is therefore given DBA authority.

```
GRANT CONNECT TO company
IDENTIFIED BY secret;
GRANT DBA TO company;
```

The tables in the database are created by the `company` user ID.

```
CONNECT USER company IDENTIFIED BY secret;
CREATE TABLE company.Customers ( ... );
CREATE TABLE company.Products ( ... );
CREATE TABLE company.Orders ( ... );
CREATE TABLE company.Invoices ( ... );
CREATE TABLE company.Employees ( ... );
CREATE TABLE company.Salaries ( ... );
```

Not everybody in the company should have access to all information. Consider two user IDs in the sales department, Joe and Sally, who should have access to the `Customers`, `Products` and `Orders` tables. To do this, you create a Sales group.

```
GRANT CONNECT TO Sally IDENTIFIED BY xxxxx;
GRANT CONNECT TO Joe IDENTIFIED BY xxxxx;
GRANT CONNECT TO Sales IDENTIFIED BY xxxxx;
GRANT GROUP TO Sales;
GRANT ALL ON Customers TO Sales;
GRANT ALL ON Orders TO Sales;
GRANT SELECT ON Products TO Sales;
GRANT MEMBERSHIP IN GROUP Sales TO Sally;
GRANT MEMBERSHIP IN GROUP Sales TO Joe;
```

Now Joe and Sally have permission to use these tables, but they still have to qualify their table references because the table owner is `company`, and Sally and Joe are not members of the `company` group:

```
SELECT *
FROM company.customers
```

To rectify the situation, make the `Sales` group a member of the `company` group.

```
GRANT GROUP TO company;
GRANT MEMBERSHIP IN GROUP company TO Sales;
```

Now Joe and Sally, being members of the `Sales` group, are indirectly members of the `company` group, and can reference their tables without qualifiers. The following command will now work:

```
SELECT *
FROM Customers
```

Notes

Joe and Sally do not have any extra permissions because of their membership in the `company` group. The `company` group has not been explicitly granted any table permissions. (The `company` user ID has implicit permission to look at tables like `Salaries` because it created the tables and has DBA authority.) Thus, Joe and Sally still get an error executing either of these commands:

```
SELECT *
FROM Salaries;
SELECT *
FROM company.Salaries
```

In either case, Joe and Sally do not have permission to look at the `Salaries` table.

See also

- *Tables Owned by Groups* on page 328

Views and Procedures Provide Extra Security

You can use views and stored procedures to tailor permissions to suit the needs of your organization.

For databases that require a high level of security, defining permissions directly on tables has limitations. Any permission granted to a user on a table applies to the whole table. There are many cases when users' permissions need to be shaped more precisely than on a table-by-table basis. For example:

- It is not desirable to give access to personal or sensitive information stored in an employee table to users who need access to other parts of the table.
- You may wish to give sales representatives update permissions on a table containing descriptions of their sales calls, but limit such permissions to their own calls.

This section describes some of the uses of views and procedures for permission management.

See also

- *View Management* on page 188

Views Provide Tailored Security

Views are useful for security when it is appropriate to give a user access to just one portion of a table.

The portion can be defined in terms of rows or in terms of columns. For example, you may wish to disallow a group of users from seeing the `Salary` column of an `Employees` table, or you may wish to limit a user to see only the rows of a table that they have created.

Example 1

The sales manager needs access to information in the database concerning employees in the department. However, there is no reason for the manager to have access to information about employees in other departments.

This example describes how to create a user ID for the sales manager, create views that provide the information she needs, and grants the appropriate permissions to the sales manager user ID.

1. Create the new user ID using the GRANT statement, from a user ID with DBA authority. Enter:

```
CONNECT "DBA"
IDENTIFIED by sql;
GRANT CONNECT
TO SalesManager
IDENTIFIED BY sales
```

You must enclose DBA in quotation marks because it is a SQL keyword, just like **SELECT** and **FROM**.

2. Define a view that looks only at sales employees:

```
CREATE VIEW emp_sales AS
SELECT EmployeeID, GivenName, Surname
FROM "DBA".Employees
WHERE DepartmentID = 200
```

   Identify the table as `"DBA".Employees`, with the owner of the table explicitly identified, so that the `SalesManager` user ID can use the view. Otherwise, when SalesManager uses the view, the **SELECT** statement refers to a table that the user ID does not recognize.

3. Give SalesManager permission to look at the view:

```
GRANT SELECT
ON emp_sales
TO SalesManager
```

   Use the same command to grant permission on a view as to grant permission on a table.

Example 2

The next example creates a view which allows the Sales Manager to look at a summary of sales orders. This view requires information from more than one table for its definition:

1. Create the view.

```
CREATE VIEW order_summary AS
SELECT OrderDate, Region, SalesRepresentative
FROM "GROUPO".SalesOrders
    KEY JOIN "GROUPO".Customers
```

2. Grant permission for the Sales Manager to examine this view.

```
GRANT SELECT
ON order_summary
TO SalesManager
```

3. To check that the process has worked properly, connect to the SalesManager user ID and look at the views you have created:

```
CONNECT SalesManager IDENTIFIED BY sales ;
SELECT * FROM "GROUPO".emp_sales ;
SELECT * FROM "GROUPO".order_summary ;
```

No permissions have been granted to the Sales Manager to look at the underlying tables. The following commands produce permission errors.

```
SELECT * FROM "DBA".Employees ;
SELECT * FROM "DBA".SalesOrders;
```

The previous example shows how to use views to tailor SELECT permissions. INSERT, DELETE, and UPDATE permissions can be granted on views in the same way.

For information on allowing data modification on views, see "Guidelines for Using Views" on page 209.

See also

- *Guidelines for Using Views* on page 190

Procedures Provide Tailored Security

While views restrict access on the basis of data, procedures restrict the actions a user may take.

A user may have EXECUTE permission on a procedure without having any permissions on the table or tables on which the procedure acts.

By default, procedures execute with the permissions of the procedure owner. For a procedure that updates a table, if the procedure owner has UPDATE permissions on the table, the user can execute the procedure. The owner of the procedure can restrict the procedure to execute with the permissions of the user executing the procedure by specifying SQL SECURITY INVOKER to CREATE/ALTER PROCEDURE statement.

See also

- *Granting Permissions on Procedures in Interactive SQL* on page 323
- *Granting Permissions on Procedures in Sybase Central* on page 323

Setting Up Task-based Security Restrictions

For strict security, you can disallow all access to the underlying tables, and grant permissions to users or groups of users to execute certain stored procedures. This approach strictly defines how data in the database can be modified.

To allow users with the particular authorities to administer the certain tasks using IQ system procedures:

1. Create a group for each desired authority.
2. Grant the authority to the designated group.
3. Grant EXECUTE permissions on the IQ procedure for performing the authority tasks to the group.
4. When a new user is created who is to be granted the authority, grant membership for this user to the group created for that authority.

Granting Users the Permissions to Run Related Stored Procedures

Since most authorities are inherited through group membership, users can inherit the authority and also the execute permissions for IQ procedures from a group.

Prerequisites

The following steps require DBA or PERMS ADMIN and USER ADMIN authority. The statement **CREATE USER USERADMIN_GRP** requires USER ADMIN authority.

To grant a user `user1`, USER ADMIN authority and permissions to execute procedures related to user administration:

Task

1. Create a group `USER ADMIN_GRP`

   ```
   CREATE USER USERADMIN_GRP
   GRANT GROUP TO USERADMIN_GRP
   ```

2. Grant USER ADMIN authority to `USERADMIN_GRP`.

   ```
   GRANT USER ADMIN TO USERADMIN_GRP
   ```

3. Grant EXECUTE permission on Sybase IQ stored procedures for user administration to `USERADMIN_GRP`.

   ```
   GRANT EXECUTE on sp_iqaddlogin
   to USERADMIN_GRP
   GRANT EXECUTE on sp_iqcopyloginpolicy
   to USERADMIN_GRP
   GRANT EXECUTE on sp_iqdroplogin
   to USERADMIN_GRP
   GRANT EXECUTE on sp_iqmodifyadmin
   to USERADMIN_GRP
   GRANT EXECUTE on sp_iqmodifylogin
   to USERADMIN_GRP
   ```

4. Grant membership in group `USERADMIN_GRP` to `user1`. The user `user1` inherits the `USER ADMIN` authority and the ability to execute the assigned IQ procedures through membership in `USERADMIN_GRP` group.

   ```
   GRANT MEMBERSHIP IN GROUP USERADMIN_GRP TO user1
   ```

Related Stored Procedures for Group Access

You may create groups that grant permissions for various related stored procedures, for example:

Group name	Grant authority	Grant EXECUTE permission on stored procedures
OPERATOR_GRP	OPERATOR	**sp_iqbackupdetails**, **sp_iqbackupsummary**, **sp_iqconnection**, and **sp_iqsysmon**
SPACEADMIN_GRP	SPACE ADMIN	**sp_iqdbspace**, **sp_iqdbspaceinfo**, **sp_iqdbspaceobjectinfo**, **sp_iqemptyfile**, **sp_iqestdbspaces**, **sp_iqfile**, **sp_iqobjectinfo**, and **sp_iqspaceused**

For a multiplex example, see Using Sybase IQ Multiplex.

How User Permissions are Assessed

Groups do introduce complexities in the permissions of individual users.

Suppose user `M_Haneef` has been granted SELECT and UPDATE permissions on a specific table individually, but is also a member of two groups, one of which has no access to the table at all, and one of which has only SELECT access. What are the permissions in effect for this user?

This is how Sybase IQ determines whether a user ID has permission to carry out a specific action:

1. If the user ID has DBA permissions, the user ID can carry out any action in the database. If the user has an authority, the user has the permissions associated with that authority.
2. Otherwise, permission depends on the permissions assigned to the individual user. If the user ID has been granted permission to carry out the action, then the action is allowed to proceed.
3. If no individual settings have been made for that user, permission depends on the permissions of each of the groups of which the user is a member. If any of these groups has permission to carry out the action, the user ID has permission by virtue of membership in that group, and the action is allowed to proceed.

   If you do not want a specific user to access a particular table, view, or procedure, then do not make that user a member of a group that has permissions on that object.

This approach minimizes problems associated with the order in which permissions are set.

See also

- *Authorities Overview* on page 305

Resources Used by Connections

Building a set of users and groups allows you to manage permissions on a database. Another aspect of database security and management is to limit the resources an individual user can use.

For example, you may wish to prevent a single connection from taking too much of the available memory or CPU resources, so that one connection does not slow down other users of the database.

Database Options that Govern User Resources

Sybase IQ provides a set of database options that the DBA can use to control resources. These options are called *resource governors*.

How to Set Options

You can set database options using the **SET OPTION** statement. For syntax, see *Reference: Statements and Options*.

Resources that can be Managed

The following database options can be used to manage resources.

- Defines the number of cursor rows to buffer.
- Limits the number of result rows from a query containing a Cartesian join.
- Sets the number of processing threads available to a connection for use in IQ operations.
- Sets the size of the cache for the IQ Temporary Store. (The server option **-iqtc** is the recommended way to set the temp cache size.)
- Limits the amount of temporary dbspace available to any one query.
- Tells the query optimizer to reject queries that might consume too many resources. If the optimizer estimates that the result set from the query will exceed the value of this option, the optimizer rejects the query and returns an error message.

The following database options affect the engine, but have limited impact on Sybase IQ:

- Sets the maximum size (in bytes) of that part of the memory that is allocated to Java applications on a per connection basis.
- Limits the number of cursors for a connection.
- Limits the number of prepared statements for a connection.

Database option settings are not inherited through the group structure.

Limits on Database Connections

You can control resources available to users with Sybase IQ Login Management, which lets you limit the number of connections to a database for all users or for an individual user, set password expirations, and lock out individual users so that they cannot connect to the database.

See also

- *User Account and Connection Management* on page 311

Using Procedures to Disable Connections

To disable connections, use the stored procedure **sp_iqmodifylogin**.

For an example, see the **sp_iqmodifylogin** procedure in *Reference: Building Blocks, Tables, and Procedures*.

Users and Permissions in System Objects

Information about the current users of a database and about their permissions is stored in the database system tables and system views.

Most system tables are owned by the special user ID SYS. It is not possible to connect to the SYS user ID.

The DBA has SELECT access to all system tables, just as to any other tables in the database. The access of other users to some of the tables is limited. For example, only the DBA has access to the SYS.SYSUSERPERM table, which contains all information about the permissions of users of the database, as well as the passwords of each user ID. However, SYS.SYSUSERPERMS is a view containing all information in SYS.SYSUSERPERM except for the password, and by default all users have SELECT access to this view. All permissions and group memberships set up in a new database for SYS, PUBLIC, and DBA can be fully modified.

User ID, Group and Permissions Information in System Tables

The following table summarizes the system tables containing information about user IDs, groups, and permissions.

All tables and views are owned by user ID SYS, and so their qualified names are SYS.SYSUSERPERM and so on.

Appropriate SELECT queries on these tables generate all the user ID and permission information stored in the database.

Table	Default	Contents
SYSUSERPERM	DBA only	Database-level permissions and password for each user ID
SYSGROUP	PUBLIC	One row for each member of each group
SYSTABLEPERM	PUBLIC	All permissions on table given by the GRANT commands
SYSCOLPERM	PUBLIC	All columns with UPDATE permission given by the GRANT command
SYSPROCPERM	PUBLIC	Each row holds one user granted permission to use one procedure

User ID, Group, and Permissions Information in System Views

The following table summarizes the system views containing information about user IDs, groups, and permissions.

View	Default	Contents
SYSUSERAUTH	DBA only	All information in SYSUSERPERM except for user numbers
SYSUSERPERMS	PUBLIC	All information in SYSUSERPERM except for passwords
SYSUSERLIST	PUBLIC	All information in SYSUSERAUTH except for passwords

View	Default	Contents
SYSGROUPS	PUBLIC	Information from SYSGROUP in a more readable format
SYSTABAUTH	PUBLIC	Information from SYSTABLEPERM in a more readable format
SYSCOLAUTH	PUBLIC	Information from SYSCOLPERM in a more readable format
SYSPROCAUTH	PUBLIC	Information from SYSPROCPERM in a more readable format

In addition to these, there are tables and views containing information about each object in the database.

Transport-Layer Security

You can secure communications between a client and the IQ server or between an IQ client and the database server using transport-layer security (TLS).

Database file encryption for Sybase IQ databases is similar to that of SQL Anywhere databases as described in *SQL Anywhere Server – Database Administration* on the Sybase Product Manuals web site. Sybase IQ also allows you to encrypt columns, as documented in *Advanced Security in Sybase IQ*.

Support of FIPS encryption, Kerberos authentication, and column encryption is included in the separately licensed Sybase IQ Advanced Security Option, which is described in *Advanced Security in Sybase IQ*.

IPv6 Support

Sybase IQ supports Internet Protocol version 6 (IPv6), which contains addressing and control information to route packets over the Internet.

IPv6 supports two^{128} unique IP addresses, which is a substantial increase over the number of addresses supported by its predecessor IPv4. Sybase IQ supports both IPv4 and IPv6 addresses anywhere you can specify an IP address on the client or server.

ODBC classes support the use of IPv6 addresses for remote data access. JDBC classes do not support the use of IPv6 addresses for remote data access.

For more information on IPv6 support, see *SQL Anywhere 11.0.1 > SQL Anywhere Server - Database Administration > Starting and Connecting to Your Database > Client/server communications > Using the TCP/IP protocol*.

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, business rules that maintain the consistency of data between tables.

Sybase IQ supports stored procedures and JDBC, which allow you detailed control over how data gets entered into the database. Procedures are discussed in *System Administration Guide: Volume 2*.

For information on JDBC, see *SQL Anywhere 11.0.1 > SQL Anywhere Server - Programming > SQL Anywhere Data Access APIs > SQL Anywhere JDBC driver*.

How Data Can Become Invalid

Here are a few examples of how the data in a database may become invalid if proper checks are not made. Each of these examples can be prevented by facilities described in this chapter.

- An operator enters orders to an orders table for a **customer_id** that does not exist in the customers table.
- An operator enters text where numeric data is required.
- An operator enters numeric data that is too wide for the column.
- A new department has been created, with **dept_id** 200, and needs to be added to the department table of the organization's database—but two people enter this information into the table.

Rules and Checks for Valid Data

To help ensure that the data in a database are valid, you need to formulate checks that define valid and invalid data and design 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. Table and column constraints can both be applied to a single column in a table. Table constraints can also set the rule for a set of columns in a table.

Constraints Should Be Built In

Constraints built into the database itself are inherently more reliable than those built into client applications, or spelled out as instructions to database users. Constraints built into the database are part of the definition of the database itself and can be enforced consistently across all applications.

Setting a constraint once, in the database, imposes it for all subsequent interactions with the database, no matter from what source. By contrast, constraints built into client applications are vulnerable every time the software is altered, and may need to be imposed in several applications, or several places in a single client application.

Because IQ data typically is entered by only a few users, and often loaded directly from other databases, IQ databases may be less vulnerable than OLTP databases to the kinds of errors that can cause invalid data, depending on which extract, transform and load process you use.

You should declare any constraints that apply, whether Sybase IQ enforces them or not. By declaring constraints, you ensure that you understand your data requirements, and are designing a database that matches the business rules of your organization.

Constraints aid IQ Optimization

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.

Here is a list of some of the types of optimization that rely on the constraints and other features you build into the database:

- Join indexes optimize queries that join data from different columns. In many cases, the join relationship for a join index relies on the foreign key constraints you specify for the tables being joined.
- **FOREIGN KEY**, **PRIMARY KEY** and **UNIQUE** column constraints and the **IQ UNIQUE** parameter can improve performance for your loads and queries.

Constraints Check Loads

Sybase IQ checks during load operations that certain constraints are obeyed:

- Sybase IQ ensures that data being loaded is the appropriate data type and length.
- If you have a join index that relies on a foreign key-primary key relationship, when synchronizing the join index Sybase IQ checks that data in the underlying tables maintains the expected one-to-many relationship between the joined columns.

See also

- *Guidelines for Creating Tables* on page 179
- *How Locking Works* on page 376
- *IQ PAGE SIZE Parameter Guidelines* on page 154
- *Number of Distinct Values* on page 297

- *Using Join Indexes* on page 223

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

To help maintain data integrity, you can use data constraints and constraints that specify the referential structure of the database.

Constraints

You can use several types of constraints on the data in individual columns or tables. For example:

- A NOT NULL constraint prevents a column from containing a null entry. 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. You could specify, for example, that salary column entries should 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 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**.

The following restrictions affect candidate keys:

- A foreign key cannot be a candidate key if a join index exists.
- You cannot specify a foreign key constraint to a candidate key that is also a foreign key.

The following integrity rules define the structure of the database:

- Keeps track of the primary keys. It guarantees that every row of a given table can be uniquely identified by a primary key that guarantees no nulls.
- Keeps track of the foreign keys that define the relationships between tables. All foreign key values either should match a value in the corresponding primary key or contain the NULL value if they are defined to allow NULL.

See also

- *Column Defaults Encourage Data Integrity* on page 344
- *Table and Column Constraints* on page 351

SQL Statements for Implementing Integrity Constraints

The following SQL statements are used to implement integrity constraints:

- This statement implements integrity constraints as the database is being created.
- This statement adds integrity constraints, or deletes constraints, from an existing database.

Column Defaults Encourage Data Integrity

Column defaults automatically assign a specified value to a particular column or set of columns whenever someone enters a new row into a database table.

The default value assigned 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.

See also

- *Data Integrity Tools* on page 343
- *Table and Column Constraints* on page 351

Supported Default Values

Sybase IQ supports the following 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

Certain column default values are not supported.

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

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 at the time a table is created, or the **ALTER TABLE** statement to add column defaults at a later time.

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. To 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. To 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. To 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.

The following statement changes the default value of a column named **order_date** from its current setting to **CURRENT DATE**:

```
ALTER TABLE sales_order
MODIFY order_date DEFAULT CURRENT DATE
```

Deleting Column Defaults

To remove column defaults, modify them to be NULL.

The following 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 Sybase IQ.

Use the following statements:

- **INSERT...VALUES**
- **INSERT...SELECT**
- **INSERT...LOCATION**
- **LOAD TABLE**
- **UPDATE**
- **SELECT...FROM...FOR UPDATE**

Sybase IQ handles defining and inserting column default values with the following requirements:

- Sybase IQ permits you to specify default values that cannot be evaluated by 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.
- Sybase IQ generates an error or warning when the server attempts to insert a default value that is not compatible 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, 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, Sybase IQ either inserts 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 is a string that contains a partial multi-byte character, then Sybase IQ may trim the partial multi-byte character before inserting the value, depending on the setting of the `TRIM_PARTIAL_MBC` database option.
- Sybase IQ generates an error message every time the server attempts to insert the default value of a column, if that default value violates the check constraint of either the table or the column.
- All constraint violations that occur during a **LOAD TABLE** operation as a result of inserting default values apply towards 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 Sybase IQ. An error is reported every time an attempt is made to insert or update the default value of a column of this type.
- Column default values defined on base tables are not propagated to joins in which these tables participate.
- Column default values are not permitted on tables that participate in join indexes and Sybase IQ generates an error, if you attempt to define a default value on such a table. This rule is similar to support for the **AUTOINCREMENT** default value.
- If a column on which a default value is defined is added to a table, then 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 in order 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 that needs 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.

See the individual sections for specific default value types later in this section for more information on defining and inserting column default values. For more information on the special values that can be used in default column value expressions, see *Reference: Building Blocks, Tables, and Procedures* .

See also

- *Explicit Data Conversions in IQ* on page 278

Column Defaults in Sybase Central

You can add, alter, and delete column defaults in Sybase Central using the Value tab of the column properties sheet.

Note: When you create a new column, some attributes are hidden until you select Data Type or Value and click the ellipses.

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.

Examples of 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 to distinguish 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.

Sybase IQ does not support default values of **UTC TIMESTAMP** or **CURRENT UTC TIMESTAMP**, nor does IQ support the database option

`DEFAULT_TIMESTAMP_INCREMENT`. 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 is an easy and reliable way of identifying 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 to prevent conflicts during information updates. These users can make a copy of tables relevant to their work on a portable computer, make changes while not connected to a multi-user 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 previous 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 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.

Sybase IQ does not support the special value **GLOBAL AUTOINCREMENT**.

Autoincrement and Negative Numbers

IDENTITY/AUTOINCREMENT is intended to work with positive integers.

The initial **IDENTITY/AUTOINCREMENT** value is set to 0 when the table is created.

Autoincrement and the IDENTITY Column

A column with the **AUTOINCREMENT** default is referred to in Transact-SQL applications as an **IDENTITY** column. Sybase IQ supports both keywords for compatibility.

The NEWID Default

UUIDs (Universally Unique IDentifiers), also known as GUIDs (Globally Unique IDentifiers), can be used to uniquely identify rows in a table.

The values are generated such that a value produced on one computer will 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 at all.

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 *Reference: Building Blocks, Tables, and Procedures > SQL Language Elements > Null value*.

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.

Functions such as **GETDATE** and **DATEADD** can be used in a constant expression default value. If the default constant expression is not a function or simple value, the expression must be enclosed 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 in the table is correct and provide information to 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 could 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 IQ 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.

See also

- *Column Defaults Encourage Data Integrity* on page 344
- *Data Integrity Tools* on page 343

UNIQUE Constraints on Columns or Tables

The **UNIQUE** constraint specifies that one or more columns uniquely identify each row in the table. If you apply the **UNIQUE** constraint, Sybase IQ enforces this condition.

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 must not contain any **NULL** values.

Example 1

The following 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 are needed to make a unique entry.

```
ALTER TABLE product
ADD UNIQUE (name, size, color)
```

See also

- *FP(3) Index* on page 206
- *Configuring FP(3) Indexes* on page 208

IQ UNIQUE Constraints on Columns

The **IQ UNIQUE** constraint specifies an estimate of the number of distinct values in a column. You can apply the **IQ UNIQUE** constraint to any column in a table. This constraint helps optimize loading of indexes.

In the Sybase Central IQ plug-in, you can add **IQ UNIQUE** constraints on the column properties page. For details, see the online help.

For example, in the `state` column of the `employee` table, you would specify **IQ UNIQUE(50)** to indicate that there are only 50 possible values (assuming U.S. states only). Each of the possible values can occur many times.

When the `MINIMIZE_STORAGE` database option is ON, it is equivalent to specifying **IQ UNIQUE(255)** on all new columns. This option is OFF by default as of Version 12.6.

CHECK Conditions on Columns

You can use a CHECK condition to specify that the values in a column must satisfy some definite criterion.

You can apply a CHECK condition to values in a single column, to specify the rules they should follow. These rules may be rules that data must satisfy in order to be reasonable, or they may be more rigid 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. Here are some examples:

Example 1

You can specify that the entry should match one of a limited number of values. For example, 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 like the following:

```
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. See the CASE clause in the **CREATE DATABASE** statement in *Reference: Statements and Options*.

Example 2

You can specify that a date or number falls in a particular range. For example, 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 the following:

```
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 used in this example has the virtue of always being 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 the column overrides the CHECK condition from the 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 )
```

Any variable name prefixed with @ could be used instead of **@col**. Any column defined using the **posint** data type accepts only positive integers unless it has a different CHECK condition explicitly specified.

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*.

Adding, Altering and Deleting Column Constraints in Sybase Central

All adding, altering, and deleting of column constraints in Sybase Central is carried out in the Constraints tab of the column properties sheet.

1. Connect to the database.

2. Click the Tables folder for that database, and click the table holding the column you wish to change.
3. Select a column and click File > Properties.

For more information, see the Sybase Central online Help.

CHECK Conditions on Columns

A CHECK condition can be applied as a constraint on the 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. This is more flexible behavior, and CHECK conditions on individual columns are recommended where possible.

Adding a Check Condition in Interactive SQL

You can add a new CHECK condition to the table or to an individual column.
For example, in a library database, the `date_returned` column for a particular entry must be later than (or the same as) the `date_borrowed` entry:

```
ALTER TABLE loan
ADD CHECK(date_returned >= date_borrowed)
```

Deleting a Check Condition in Interactive SQL

You can delete a CHECK condition from an individual column.

To delete a CHECK condition on a column, set it to NULL.

The following statement removes the CHECK condition on the **phone** column in the **customer** table:

```
ALTER TABLE customer MODIFY phone
CHECK NULL
```

Replacing a Check Condition in Interactive SQL

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.

The following 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 of modifying a CHECK condition defined on the table, as opposed to a CHECK condition defined on a column.

- 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 on Tables Using Interactive SQL

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

The relational structure of the database enables the database server to identify information within the database. Sybase IQ also ensures that primary key-foreign key relationships between tables are properly upheld by all the rows in any join index relying on these relationships.

See also

- *Star Joins* on page 234
- *Creating Foreign Keys* on page 187
- *Foreign Key Creation* on page 357

How to Declare Entity Integrity

Once you specify the primary key for each table, no further action is needed by client application developers or by 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 modified at a later date, the primary key may 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 enter the **CREATE TABLE** or **ALTER TABLE** command explicitly: the application generates the statement itself from the information you provide. See *Reference: Statements and Options*.

How to Enforce Entity Integrity

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 *Introduction to Sybase IQ > About Sybase IQ > Demo Database > Table Names and Owners* 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 items ordered. One column contains an **id** specifying 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 which line corresponds to the item. The two columns **id** and **line_id**, taken together, 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 would be 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.

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. Sybase IQ also optionally rejects rows that violate data integrity as specified by the user.

Referential Integrity

For the foreign key relationship to be valid, the entries in the 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**.

Foreign Key Creation

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.**

See also

- *Entity and Referential Integrity* on page 355
- *Star Joins* on page 234
- *Creating Foreign Keys* on page 187

Foreign Key Example

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. 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 could reference a column in the department table containing duplicate entries, there would be no way of knowing which row in the department table is the appropriate reference. This is a mandatory foreign key.

Referential Integrity Violations

Sybase IQ supports referential integrity with RESTRICT action (the ANSI default) at the statement level.

This means that Sybase IQ denies requests for updates and deletes on the primary key or column(s) with a unique constraint that removes any value upon which correspondent foreign key(s) depend. (You must be careful about the order in which you request deletes and updates.) Sybase IQ issues an error message and rolls back load operations that violate referential integrity, but lets you specify that certain rows be ignored.

See also

- *How to Disable Referential Integrity Checking* on page 363

Enforcing Referential Integrity with Existing Unenforced Foreign Keys

You can enforce referential integrity with unenforced foreign keys.

1. Identify the candidate key to foreign key relationship.

   In the schema in the Foreign Key Example, 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 the unenforced foreign key constraint via the **ALTER TABLE** statement if one exists.

   For example:

   ```
   ALTER TABLE DEPT DROP FOREIGN KEY MGR_EMPNO;
   ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPT;
   ```

   In the preceding schema, we need to drop unenforced foreign key constraints MGR_EMPNO and EMPLOYEE(DeptNo) referencing DEPT(DeptNo). 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 Integrity in a New Table

When creating a new table, enforce referential integrity as follows:

To enforce referential integrity:

- Create the primary table, for example:

  ```
  CREATE TABLE DEPT(DeptNo int primary key,
  DeptName varchar(20),
  Mgr int );
  ```

- 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);
  ```

- Add the foreign key constraint. For example:

  ```
  ALTER TABLE DEPT ADD FOREIGN KEY MGR_EMPNO(Mgr) REFERENCES
  EMPLOYEE(EmpNo);
  ```

Example 1

Another way to create the foreign table follows. 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));
```

To drop a foreign key constraint.

- When there is no role name assigned, as in the first **CREATE TABLE** example, the default role name for the specified foreign key is DEPT:

  ```
  ALTER TABLE EMPLOYEE DROP FOREIGN KEY DEPT;
  ```

  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*.

- In the second **CREATE TABLE** example, the role name EMP_DEPT was assigned, so you must specify it when dropping the key, as follows:

  ```
  ALTER TABLE EMPLOYEE DROP FOREIGN KEY EMP_DEPT;
  ```

Example 3

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
```

Example 4

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;
```

Example 5

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. >

Loss of Referential Integrity

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 would 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 would violate referential integrity.

Sybase IQ provides protection against both types of integrity loss.

When a referenced candidate key is updated or deleted, Sybase IQ disallows **UPDATE** or **DELETE**.

Concurrent Operations

The referential integrity feature of Sybase IQ restricts concurrent updates or deletes on a primary table during loads or inserts on a foreign table.

Table 43. Concurrent operations that return an IQ 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	to **DELETE** its associated primary table with deletable row(s).
	to **UPDATE** its associated primary table.
	to **TRUNCATE** its associated primary table.

Sybase IQ also generates an error for a request by one transaction to **ALTER TABLE ADD** foreign key or **DROP** foreign key while there are old version(s) of foreign table and/or primary table in use by other transactions.

For both enforced and unenforced foreign key and primary key, Sybase IQ allows:

- Simultaneous **LOAD/INSERT** on one or more foreign tables and the shared primary table.
- Simultaneous **LOAD/INSERT** on foreign table(s) and **DELETE/UPDATE/TRUNCATE TABLE** on another one or more foreign table(s).
- Simultaneous **DELETE/UPDATE/TRUNCATE TABLE** on 2 or more foreign tables, even if sharing the same primary table.
- Simultaneous **DELETE/TRUNCATE TABLE** on foreign table(s) and **DELETE/UPDATE/TRUNCATE TABLE** on 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 Sybase IQ guarantees consistent referential integrity checks while allowing concurrent **LOAD/INSERT/UPDATE** operations on the foreign table and **LOAD/INSERT** operations on the primary table.

Sybase IQ also verifies that deleted old values do not exist in a foreign table when a transaction requesting **DELETE** or **UPDATE** starts. This provides consistent referential integrity checking during concurrent **DELETE** on a foreign table and **DELETE/UPDATE** on a PRIMARY Table.

To understand concurrent operations on foreign and primary tables, 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 foreign table and primary table should be allowed and which return an error; the table applies only to enforced foreign keys and candidate key.

Table 44. 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	Not Applicable	Not Applicable	Not Applicable	Not Applicable	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 concurrency conflict if one transaction loads foreign key columns while another updates non-associated 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 the read locks on the tables are released, you cannot alter or drop the tables whose foreign keys are used as join conditions or grouping columns. You can, however, load, insert, delete, and update these tables while the query is running.

How to Disable Referential Integrity Checking

You can use the Sybase IQ option `DISABLE_RI_CHECK` to bypass referential integrity checking if desired.

Because bypassing referential integrity checking defeats the purpose of having the feature, Sybase recommends that you use this option carefully.

See also

- *Referential Integrity Violations* on page 358

Integrity Rules in the System Tables

All the information about integrity checks and rules in a database is held in the following 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.

For a description of the contents of each system table, see *Reference: Building Blocks, Tables, and Procedures*. You can use Sybase Central or **dbisql** to browse these tables and views.

Transactions and Versioning

The Sybase IQ approach to transaction processing, called snapshot versioning, has implications for performance and other aspects of database administration.

See also

- *Managing User IDs and Permissions* on page 303
- *Permissions for Modifying Data* on page 243
- *Security Overview* on page 4
- *Intermediate Versioning* on page 17
- *Table Versioning Controls Access to Join Indexes* on page 238

Transactions and Versioning Overview

Transaction processing ensures that logically related commands execute as a unit.

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 Sybase IQ approach to transaction processing, called *snapshot versioning*, supports the highest level of isolation recognized by ISO.

Transactions

Transactions are groups of SQL statements.

Each transaction performs a task that changes your database from one consistent state to another. These units play an important role in protecting your database from media and system failures, and in maintaining the consistency of your data.

Logical Units of Work

A transaction is a logical unit of work.

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. In other words, Sybase IQ executes all the statements within a transaction as a unit. At the end of each transaction, changes can be committed to make them permanent. If for any reason any of the commands in the transaction do not process properly, then 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 break the work of each user into 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.

Note: Sybase IQ processes transactions quite differently from the way SQL Anywhere does when it operates without IQ. This chapter describes how Sybase IQ handles transactions. If you are working in an Anywhere-only database, see *SQL Anywhere 11.0.1 > SQL Anywhere Server - SQL Usage > Creating Databases > Using transactions and isolation levels > How locking works* for information on transactions and locking.

Commands and Transactions

Sybase IQ allows commands to be grouped into transactions.

In most cases, 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.

Events that Start Transactions

Certain events can start transactions automatically.

Transactions start automatically with one of the following events:

- The first statement following a connection to a database.
- The first statement following the end of a previous transaction.

Sybase IQ also supports Transact-SQL commands, such as **BEGIN TRANSACTION**, for compatibility with Adaptive Server Enterprise. IQ allows you to explicitly start a transaction using the **BEGIN TRANSACTION** command.

Events that End Transactions

Certain events complete transactions.

Transactions complete with one of the following 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.
- A statement with a side effect of an automatic commit is executed.

Database definition commands, such as **ALTER**, **CREATE**, and **DROP** all have the side effect of an automatic commit. You can also use two **dbisql** options to cause a commit to occur automatically.

Data Caches and Page Updates

Reading from and writing to the cache reduces the number of number of times Sybase IQ must access the disk.

When you execute a write operation, Sybase IQ does not immediately write the data to disk. Instead, it writes it into a data *cache*, an area in memory where it stores pages from the database while they are in use. This is an essential part of IQ's high performance.

Eventually, IQ must write dirty pages—that is, pages that have been updated—to the disk. Sybase IQ writes dirty pages to disk each time a transaction commits. This approach is a major benefit to IQ users, because it means that IQ does not need to log data insertions in the transaction log. By not logging the very large insertions that are typical with IQ, users gain tremendous savings in disk and performance cost.

Transaction Subdivisions

You can identify important states within a transaction and return to them selectively or cause other actions to occur by using savepoints.

See also

- *Savepoints Within Transactions* on page 383

Transaction Activity Snapshots

The **sp_iqtransaction** stored procedure displays a snapshot of transaction activity, such as main and temporary space created and in use, open cursors, and savepoints.

It returns a row for each transaction control block in the IQ transaction manager.

Concurrency Overview

Special mechanisms within the database server allow Sybase IQ transactions to execute concurrently without interfering with each other.

How Sybase IQ Concurrency Works

While executing the SQL statements that comprise one transaction, the database server can execute some or all of the statements in other transactions.

Sybase IQ's approach to concurrency is designed for the data warehouse. Typically, in a data warehouse environment, many users need to read from the database, but only the DBA needs to update it. However, there is often a need to be able to make those updates while other users continue to request and receive query results.

Sybase IQ allows many simultaneous connections by many users to one database. It can also process transactions from more than one connected user or application concurrently.

Sybase IQ ensures that all database operations occur within a transaction, and that these operations do not interfere with each other. It does so by setting access restrictions at the table level, and by using a technique called snapshot versioning. On a given table, IQ allows concurrent processing of multiple read transactions, but only one write transaction. This approach maintains the internal consistency of the database.

See also

- *Versioning Overview* on page 368

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.

Benefits of Concurrency

Sybase IQ's approach to concurrency gives query users immediate access to information, and allows you to ensure the safety and accuracy of the information they receive.

A data warehouse is a common repository of information shared by a large number of people. These people may need frequent access to the information. To avoid impeding their work, the database server must be able to process many transactions at the same time.

Moreover, many sites also require frequent updates to the database. In high availability sites, the DBA cannot postpone insertions and deletions to a time when exclusive access is possible. Similarly, it is important to be able to back up the database on a regular basis, without disrupting the activities of other users.

Versioning Overview

Sybase IQ uses snapshot versioning to allow transactions to operate concurrently.

You can think of snapshot versioning as you would a snapshot you take with a camera. When you photograph a snapshot of an object or scene, you get an image of it as it appears at a given moment in time. Likewise, when IQ takes a snapshot of an object in your database, it retains an image of that object at a given instant in time.

Unlike a camera, though, IQ does not need to make a copy of the entire object each time the image changes. Instead, it copies only the parts of the image—the database pages—that have changed. Database pages that have not changed are shared among all active versions in the database.

IQ takes its snapshot when the first command is executed following a connect, commit, or rollback. The user can force the snapshot to be taken earlier by executing an explicit **BEGIN TRANSACTION** command. Throughout the transaction, a user who reads from the object sees the unchanged image, or snapshot version.

See also

• *How Sybase IQ Concurrency Works* on page 367

Table-level Versioning

With table-level versioning, Sybase IQ can control access to the data at the level where write operations occur, and where query results are focused.

In Sybase IQ, at the user-visible level, the unit of versioning is the table.

Table-level versioning makes sense for Sybase IQ for these reasons:

- IQ data structures aggregate data for columns at the table level.
- Most IQ insertions and deletions write data table-wide.

Internally, however, data is versioned at the page level. This approach helps conserve system resources.

A given IQ table may consist of millions of pages of data. When you update that table, you may be writing to only a small percentage of those pages. It would require a vast amount of disk space to maintain a complete copy of each version of an entire table. Sybase IQ saves on disk space by allowing table versions to share pages that are not being updated.

Sybase IQ's table level versioning is extended for its multiplex databases. When a transaction creating a new version of a table commits on a write server, the control information describing that new version is available to all the secondary servers instantly. New transactions beginning on the secondary server automatically see these new versions of the tables, just as new transactions on the write servers do.

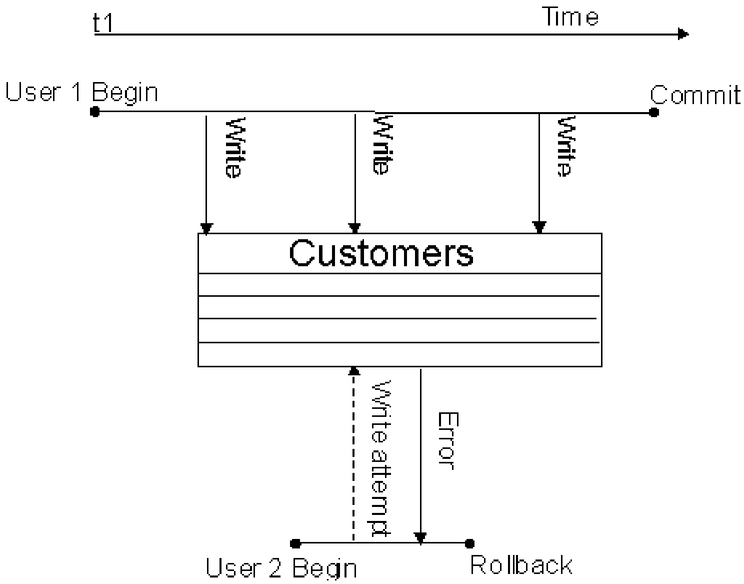
One Writer and Multiple Readers at the Table Level

On a given table, IQ permits only one user to have write access for doing insertions and deletions, and multiple readers to issue queries concurrently.

Imagine a situation such as the one shown in this figure. First, User 1 begins a transaction and starts to insert data into the `customer` table. As long as User 1's transaction remains open, no other user can write to the `customer` table. Any transaction that attempts to write to the `customer` table receives an error until User 1's transaction commits.

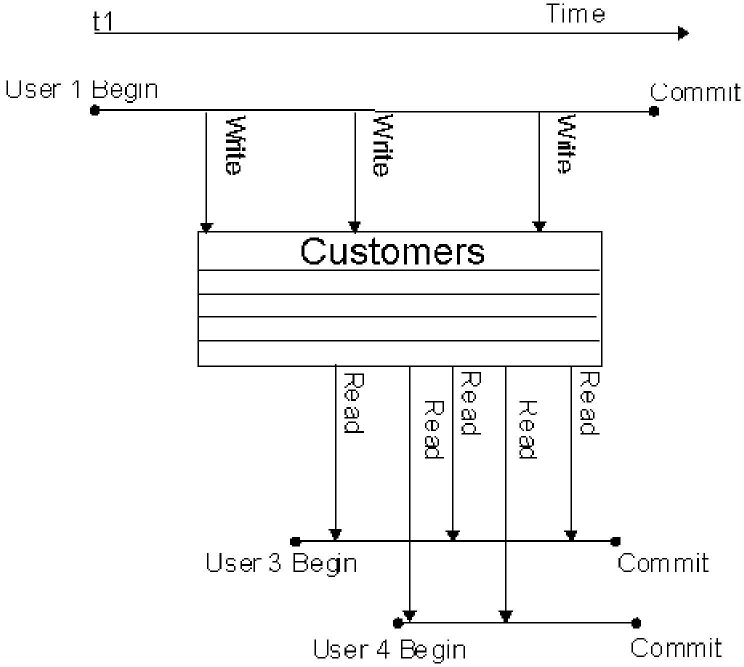
In the figure, User 2 gets an error for attempting to write before User 1's transaction commits. User 2's application determines whether to roll back the transaction, or to try writing to a different table. However, User 2 cannot write to the `customer` table again in the same transaction.

Figure 2: Only one writer at a time



Meanwhile, other users can read from the `Customers` table at any time. In this way queries can proceed while the database administrator inserts and deletes table data. In the following figure, User 3 and User 4 are able to query the `Customers` table while User 1's write transaction remains open.

Figure 3: One writer, multiple readers



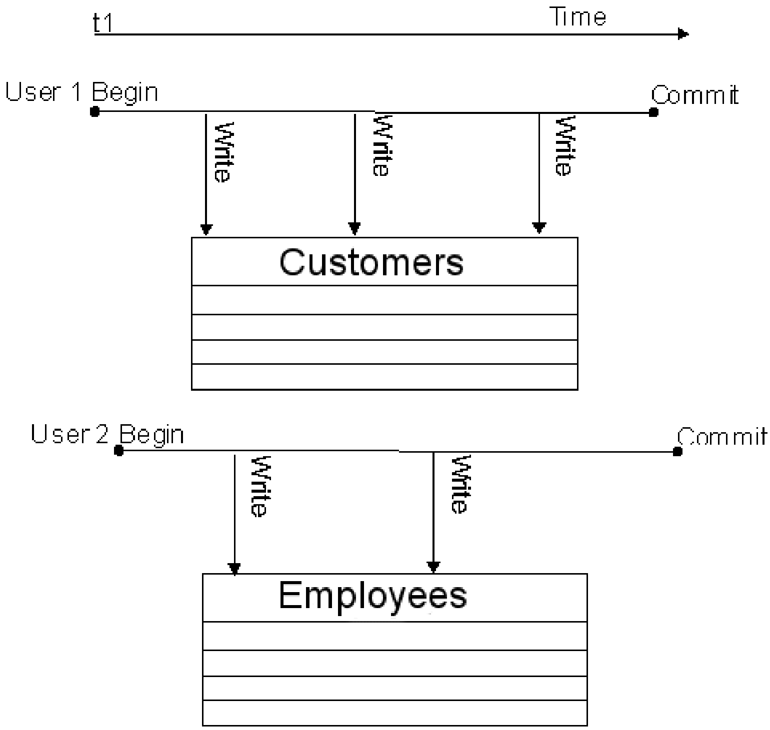
Multiple Writers and Readers in a Database

Within an IQ database, multiple read-only and read-write users can operate concurrently, as long as the writers are inserting data into (or deleting it from) different tables.

For example, while User 1's transaction is inserting and deleting in the `Customers` table, User 2 can begin a transaction that loads data into the `Employees` table, as shown the figure. At the same time other users can execute transactions that issue queries to both of these tables, or to any other tables in the database.

In general, read-only users connect to any secondary server and read-write users connect to the write server. Read-write users may also connect to query servers, but can only modify local data in global or temporary tables and SQL Anywhere base tables.

Figure 4: Concurrent Insertions to Different Tables



Data definition operations on a single table lock out all other readers and writers from that table.

See also

- *Locks for DDL Operations* on page 378

Transactions Use Committed Data

Every transaction uses the latest committed version of the database as of the time the transaction begins.

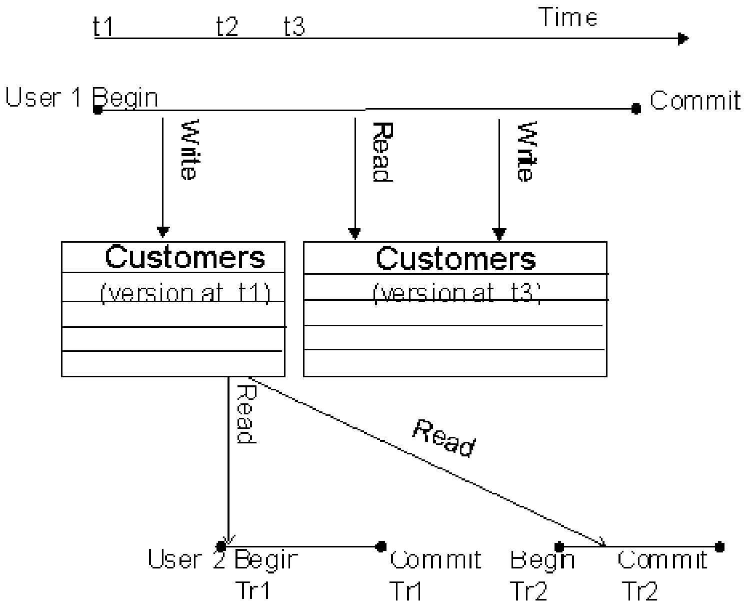
It uses that version until the transaction commits.

Committed data results when a write transaction commits.

The time a transaction begins is called its Start Timestamp. The start timestamp can be any time before the transaction's first read. Any insertions and deletions the transaction makes are reflected in the snapshot. Thus, for the user executing a transaction, the image in the snapshot changes whenever that transaction writes data to the table, and then reads it again. For all other users, the image remains static until their transaction commits.

In other words, every transaction begins with a snapshot of the data in a reliable state. The snapshot of the data that you see when you issue a query does not change, even if another user is updating the table you are reading. For example, in the following figure, when User 1's write transaction begins, it uses the `customer` table version that was committed most recently. User 2's transaction begins after User 1 has begun writing, but before User 1 commits. Therefore, User 2's first transaction (Tr1) does not see any of User 1's updates. User 2's second transaction begins after User 1 commits, so it sees all of User 1's changes.

Figure 5: Transactions Use Committed Data



The data that a writer sees changes only according to the changes he or she makes; no other transaction can change what a writer sees until the writer's transaction commits. For example, in the figure, User 1 inserts some data, then does a query, and then deletes some data. Those query results reflect the insertions that User 1 has just made.

Other transactions that begin after User 1's transaction begins but before it commits see the version of the data from the time User 1's transaction begins. They can't see the latest changes, because those changes were not yet committed. As soon as User 1's transaction commits, new transactions see User 1's changes.

See also

- *Timing of Commit Operations on Read Transactions Affects Versions* on page 374

Timing of Commit Operations on Read Transactions Affects Versions

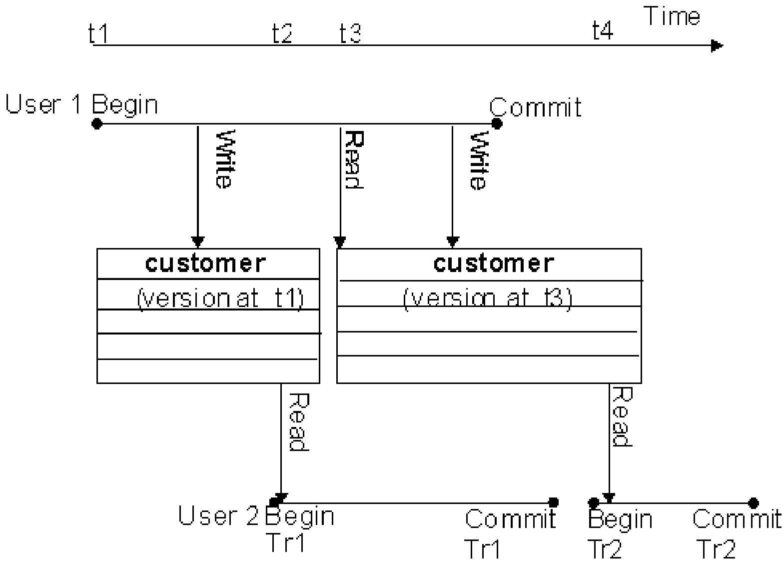
While a read transaction cannot affect what an existing write transaction sees, committing a read transaction does have implications for other transactions.

- If a user's read transaction commits before a concurrent write transaction does, and that user begins a new read transaction, the version remains the same.
- If a read transaction commits after a concurrent write transaction does, any new transaction, whether read-only or read/write, uses a new version.

The *Transactions use committed data* figure is an example of the first instance. Both of User 2's transactions use the same version as User 1's transaction began with, because that is the latest committed version of the data.

The following figure shows what happens in the second instance. This time, User 2's first read transaction (Tr1) commits after User 1's write transaction. When User 2's second transaction (Tr2) begins, it uses a new version that reflects the committed data from User 1.

Figure 6: Effect of Read Transaction Committing



See also

- *Transactions Use Committed Data* on page 372

Hold Cursors Span Transactions

The only exception to the rule that transactions always use the latest committed version is in transactions that use hold cursors.

Hold cursors are treated differently because they can span transactions.

See also

- *Cursors in Transactions* on page 388

How Sybase IQ Tracks Versions

Sybase IQ assigns a version identifier to each database object that exists in the metadata, and that has a life span beyond a single command.

IQ uses these version identifiers to ensure that writes to any database object are always based on the latest version of the object. It keeps each active version of a database object on disk.

When an older version is no longer needed by active transactions, Sybase IQ removes it from the cache. A version is needed until the transactions using it do one of the following:

- Commit
- Roll back
- Issue a **RELEASE SAVEPOINT** command releasing that version

In addition, for non-multiplex databases, Sybase IQ recognizes when no other transaction can use a particular version of a table and frees that space sooner than waiting for the oldest active transaction to commit or roll back. You are most likely to benefit from this feature if you are doing large numbers of small inserts, deletes, and updates.

See also

- *Savepoints Within Transactions* on page 383

Versioning of Temporary Tables

A temporary table that is created in the database is called a *global 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.

A *local temporary table* is declared rather than created in the database. 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, 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. Be sure to commit the data in the local temporary table before creating an index. If you attempt to create an index using uncommitted data, you may get the following error message: “Local temporary table, *<tablename>*, must be committed in order to create an index.”

For purposes of versioning, Sybase IQ makes no distinction between base tables (main database tables) and 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.

See also

- *Data Storage* on page 161
- *Dbspace Management Example* on page 172
- *Disk Space* on page 16
- *Locks for DDL Operations* on page 378
- *Disconnecting All Connections from a Database in Interactive SQL* on page 159

Versioning Prevents Inconsistencies

Without versioning, concurrent read and write operations could cause inconsistencies in the database.

The table-level versioning provided by Sybase IQ prevents inconsistencies both by *serializing* transactions, and by making the table the version level.

Sybase IQ 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. With this method, IQ takes on full responsibility for preventing inconsistencies.

While any transaction processing system is designed to ensure that the database remains consistent, the Sybase IQ approach means that users don't need to worry about placing their queries and updates in appropriate transactions. IQ begins and ends transactions automatically, and ensures that read and write operations do not interfere with each other.

How Locking Works

All Sybase IQ locks occur automatically, based on the type of operation a user requests.

You do not need to request a lock explicitly. The transaction that has access to the table is said to hold the lock.

When a table is locked in Sybase IQ, no other transaction can have write access to it, but any transaction can have read access to it. Data definition operations form an exception to this universal read access; see the discussion below for details. Any other write transaction that attempts to access a table with a write lock on it receives an error.

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. For syntax, see *Reference: Statements and Options > SQL Statements > LOCK TABLE Statement*.

See also

- *Cannot Write to a Locked Table* on page 506
- *Interactive SQL* on page 132
- *Managing Write Lock Contention on a Table* on page 506
- *Guidelines for Creating Tables* on page 179
- *IQ PAGE SIZE Parameter Guidelines* on page 154
- *Number of Distinct Values* on page 297
- *Rules and Checks for Valid Data* on page 341

Locks for DML Operations

Data manipulation language (DML) operations include insertions, deletions, updates, and queries. For all such operations, Sybase IQ permits one writer and multiple readers on any given table.

This rule has the following implications:

- 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 a single 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 will fail. The transaction can continue, but only with read operations or with writes to other tables.

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 at one time.

To avoid future version errors from subsequent DML statements, reserve a WRITE lock on the table or set of tables that you plan to modify. See the **LOCK TABLE** statement in *Reference: Statements and Options*.

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. If **SYNCHRONIZE JOIN INDEX** fails due to `table x has no data with which to join the other tables`, all database tables that participate in the join indexes and join virtual tables remain locked in WRITE mode until you explicitly disconnect or issue a **COMMIT** or **ROLLBACK** statement. Explicit **COMMIT** or **ROLLBACK** is also required to release locks when DML statements fail for example, 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, Sybase IQ returns SQL Anywhere Error -210.

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 the following 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.

See also

- *Data Storage* on page 161
- *Dbspace Management Example* on page 172
- *Disk Space* on page 16
- *Versioning of Temporary Tables* on page 375
- *Disconnecting All Connections from a Database in Interactive SQL* on page 159
- *Multiple Writers and Readers in a Database* on page 371

DDL Locking Errors

Errors may occur if you issue a DDL command when another DDL command is in process.

If more than one DDL command is attempted at the same time, users may get 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 the message:

```
Run time SQL Error
```

If a **CHECKPOINT** command is in progress, a user who issues a **CREATE DBSPACE** or **DROP DBSPACE** command gets the following message:

```
Cannot perform requested command as there is a
CHECKPOINT command in progress.
```

A user who issues **CREATE DBSPACE** during a drop gets the message:

```
Cannot perform requested command as there is a
DROP DBSPACE command in progress.
```

A user who issues **DROP DBSPACE** during a create gets the message:

```
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 will fail.

If a transaction modifies the definition of a table that is part of a join index, it locks every table with any columns that are joined in that index. This result occurs whether or not the particular columns in the original write transaction are being joined.

There is an exception to these rules for index creation commands. **CREATE INDEX** and **CREATE JOIN INDEX** can occur concurrently with a **SELECT** on the table(s) affected by the index creation. Sybase IQ prevents use of the new index or join index until the transaction creating the index commits.

While the commands **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. See *Reference: Statements and Options* for details of setting options.

Primary Keys and Locking

Because only one user can update a table, primary key generation does not cause concurrency conflicts.

Tools for Managing Locks

While locking and unlocking occurs automatically, Sybase IQ helps you manage locks by means of stored procedures, the IQ monitor, and database and server options.

Displaying Active Locks

An attempt to write to a table fails if another transaction holds a lock on that table.

To identify the user who has a table locked:

1. Run the **sp_iqtransaction** procedure.
2. Find the transaction identifier in the output of **sp_iqtransaction**.
3. Look for the name of the user in the same row of output.

Example

For example, when another transaction holds a lock, this message displays:

```
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.

Information about Locks in sp_iqlocks

The **sp_iqlocks** procedure displays information about locks currently held in the database.

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

The **sp_iqtransaction** procedure provides more detailed information about transactions. See *Reference: Building Blocks, Tables, and Procedures*

Tools for Investigating Lock Contention

Some load or query performance issues result from lock contention.

To find out if lock contention may be affecting performance on your system, use the facilities provided by IQ or your operating system:

- Run the 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.

If your kernel system time is greater than 10%, you may be experiencing lock contention.

Sybase IQ limits lock contention by partitioning your IQ main and temporary caches. The default level of partitioning is based on the number of CPUs on your IQ server, and should be adequate under most conditions. If you suspect lock contention, you may find it useful to control the level of partitioning directly by setting either the **-iqpartition** server startup option or the **Cache_Partitions** database set option.

Note: Higher than normal kernel system time can also indicate that your kernel is not well tuned. If this is the case, you probably need to adjust kernel parameters; changing IQ settings will not overcome an improperly tuned kernel.

See also

- *Data Loads* on page 15
- *Tune Bulk Loading of Data* on page 297
- *Bulk Loads with the LOAD TABLE Statement* on page 254
- *Load Time Environment Adjustments* on page 298

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. IQ snapshot versioning supports the highest level of isolation. At this level, all schedules may be serialized.

Snapshot versioning maintains this high level of isolation between concurrent transactions by following these rules:

- Transaction management maintains a snapshot of committed data at the time 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 Sybase IQ provides prevents several types of inconsistencies. The ones most commonly encountered are listed here:

- **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 Update –** 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.

Sybase IQ protects you from all of these inconsistencies by ensuring that only one user can modify a table at any given time, by keeping the changes invisible to other users until the changes are complete, and by maintaining time-stamped 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) using **SET OPTION ISOLATION_LEVEL**, 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*.

Checkpoints, Savepoints, and Transaction Rollback

Besides permitting concurrency, transaction processing plays an important role in data recovery. Database recovery always recovers every committed transaction. Transactions that have not committed at the time of a database failure are not recovered.

Sybase IQ relies on three transaction-related commands that help you recover a stable set of data in the event of system or media failure. These commands set checkpoints, set and release savepoints, and roll back transactions.

Checkpoints

A checkpoint marks a significant point in a transaction, when Sybase IQ writes to disk certain information it tracks internally. IQ uses this information in the event you need to recover your database.

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.

Sybase IQ is an OLAP database. A single OLAP transaction can change thousands or millions of rows of data. For this reason, Sybase IQ 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.

Checkpoints Aid in Recovery

In order to recover from a system or media failure, Sybase IQ must be able to restore the database to a point of internal consistency.

IQ uses checkpoints to generate reference points and other information needed to recover databases. The information that IQ writes to disk at each checkpoint is essential to the recovery process.

When Checkpoints Occur

Most Sybase IQ checkpoints occur automatically. You can also set explicit checkpoints, although you do not need to do so.

A checkpoint occurs at the following times:

- 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. It is set by default at 60 minutes. To adjust the checkpoint interval, use the **SET OPTION** statement. See *Reference: Statements and Options*. Adjusting the checkpoint time or issuing explicit checkpoints may be unnecessary. Controlling checkpoints is less important in Sybase IQ than in OLTP database products, because IQ writes the actual data pages after each transaction commits.

See also

- *How Transaction Information Aids Recovery* on page 386

Savepoints Within Transactions

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 disk—to less than a full transaction's worth. Setting savepoints allows you to subdivide transactions.

You can undo all changes after a savepoint using a **ROLLBACK TO SAVEPOINT** statement.

See also

- *Transaction Subdivisions* on page 367
- *How Sybase IQ Tracks Versions* on page 375

Release of Savepoints

Once a **RELEASE SAVEPOINT** statement executes or the transaction ends, you can no longer use the savepoint.

Releasing a savepoint frees up the version pages that have been used, up to that savepoint. Remember that data is versioned at the page level internally. Sybase IQ maintains a separate copy of just the updated pages; the remaining pages are shared with the previous version. By releasing savepoints, you free up the pages associated with them, and thus make better use of your disk space.

Releasing savepoint *n* both releases all resources after that savepoint, and gives up your ability to roll back to any intermediate savepoints.

No locks are released by the **RELEASE SAVEPOINT** command.

Rollbacks to Savepoints

You can undo all changes after a savepoint by issuing a **ROLLBACK TO 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 that you have a series of savepoints in a transaction, and then perform a write operation. You then 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.

Sybase IQ supports savepoint operations on updatable cursors.

See also

- *Cursors in Transactions* on page 388
- *Effect of Rollback* on page 385

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. If you want to retain data page versions beyond the end of a single DML command, you need to set your own, named savepoints.

Named and Nested Savepoints

Named, nested savepoints provide many active savepoints within a transaction.

Changes between a **SAVEPOINT** and a **RELEASE SAVEPOINT** can still be canceled 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 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, you undo all of the operations in that transaction.

Rolling back the database means returning the database to an earlier state.

What Causes a Rollback

Rollbacks can occur either due to an explicit user request, or automatically.

You use a **ROLLBACK** statement to undo any changes to the database since the last **COMMIT** or **ROLLBACK**.

You use a **ROLLBACK TO SAVEPOINT**statement to undo any changes to the database since the **SAVEPOINT** you name, or else to the last **SAVEPOINT**.

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.

It also releases locks:

- Transaction rollback releases all locks held by the transaction.
- Rollback to a savepoint releases all locks acquired after that savepoint.

Rollback of open cursors deletes all cursor information and closes both hold and non-hold cursors:

- Transaction rollback closes all cursors. It does not matter 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.

See also

- *Cursors in Transactions* on page 388
- *Rollbacks to Savepoints* on page 384
- *Cursors and Versioning* on page 389

System Recovery

In the event of a system failure or power outage, or when you restart the database server after it has been stopped, Sybase IQ attempts to recover automatically.

During 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, 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, you need to restart the IQ server with at least as many connections as were actually in use when the failure occurred. Temporary table contents are not recoverable.

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.

Sybase recommends that you 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.

See also

- *Data Protection* on page 405
- *Operating System Session Shutdown* on page 42
- *Resource Issues Running sp_iqcheckdb* on page 461
- *System Recovery and Database Repair* on page 453
- *Validating Your Database* on page 426

How Transaction Information Aids Recovery

The Sybase IQ recovery mechanism is designed for the data warehouse. Typically in this environment, few transactions occur, but each transaction can be quite time consuming.

To best suit this model, Sybase IQ performs database updates by making them on a copy of the actual database page, and then writes the data to disk whenever a write transaction commits. It also records the following information:

- The location and quantity of changed data for each transaction. It stores this information in a *transaction log*.
- The location of any version pages and free space on disk. It uses this information to free up space when versions are no longer needed. All versions created throughout the duration of 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, instead of repeating all of the lengthy transactions that have occurred, 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, the transaction log continues to grow in size. In systems with a high number of transactions that change data, the log can grow to be very large over a period of time, requiring periodic truncation of the log.

The checkpoint information is deleted at the next checkpoint. Information related to particular savepoints is deleted when the savepoint is released or rolled back.

See also

- *When Checkpoints Occur* on page 383
- *Running Backups* on page 412

Performance Implications of Snapshot Versioning

Snapshot versioning should have a minimal impact on performance. The flexibility you gain by being able to update the database while other users read from it far outweigh any negative effects. There are certain resource issues you should be aware of, however:

- 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. See also the bullet on disk space.
- 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 can block other read or write transactions only very briefly.

Remember that all of these performance and disk use factors only affect your system in the degree to which you take advantage of IQ's 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

In order to delete data, you may actually need to increase disk space by adding a dbspace to your IQ 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. IQ needs to 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 happen to be distributed across many data pages, then you need space in your IQ store to retain all of those extra data pages.

For example, assume that you need to delete ten rows from a database where each page holds 100 rows. If each of those ten rows is on a separate data page, then your IQ Store needs to have space for ten version pages, each big enough to hold 100 rows. While this distribution is unlikely, 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, Sybase IQ halts the deletion and displays 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 need normally that much space in your database, you can drop the dbspace to regain the extra disk space for other purposes. Be sure you do so before inserting any data which might need to use 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.

See also

• *Disk Space Usage* on page 203

Cursors in Transactions

A cursor allows you to return the results of a **SELECT** in the form of a data type called a cursor.

A cursor is similar to a table, but has the additional property that 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.

Cursors are of most use when you program procedures, or when you write applications that access a database using Embedded SQL. They are also used by many front-end query tools. They are not available when using **dbisql** interactively.

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 can even be inconsistent. For this reason, you will want to impose an order by appending an **ORDER BY** phrase to your **SELECT** statement.

The **sp_iqcursorinfo** stored procedure displays information about cursors currently open on the server. See *Reference: Building Blocks, Tables, and Procedures* .

See also

- *Hold Cursors Span Transactions* on page 374
- *Effect of Rollback* on page 385
- *Rollbacks to Savepoints* on page 384

Cursors and Versioning

When you use cursors, Sybase IQ needs to 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 then update that table later on at time *y*. Sybase IQ needs to retain the version of the `customer` table from time x until you are done using `cust_cursor`.

The support of cursors by Sybase IQ is oriented toward their likely use in DSS applications. The following sections discuss specific cursor characteristics with implications for transaction processing.

See also

- *Effect of Rollback* on page 385

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 asensitive 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. Sybase IQ supports asensitive updatable cursors.

Cursor Scrolling

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** to **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 will be faster with this option set to **OFF**.

If your front-end application rarely performs backward-scrolling, make **FORCE_NO_SCROLL_CURSORS = 'ON'** a permanent **PUBLIC** option. It will use less memory and improve 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 Sybase IQ, hold cursors are updatable until they are committed. After the commit, the 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, with long transactions, 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.

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.

Sybase IQ supports positioned fetches, which can be helpful in long query transactions. Sybase IQ also supports positioned update and delete operations, which are intended for shorter insertions and deletions. For the most part, updates to 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), not 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.

How to Control Message Logging for Cursors

By default, cursor operations are not logged in the IQ message file.

If you need to track cursor operations in order 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.

International Languages and Character Sets

Configure your Sybase IQ installation to handle international language issues.

When you create a database, you specify a collating sequence or *collation* to be used by the database. A collation is a combination of a *character set* and a *sort order* for characters in the database.

The database collation is used to sort and compare all character data types in the database, including object names, such as table and column names.

Sybase IQ support of database collations takes advantage of the space efficiency and speed of the SQL Anywhere Collation Algorithm.

- The database option `SORT_COLLATION` allows implicit use of the **SORTKEY** function on **ORDER BY** expressions. When the value of this option is set to a valid collation name or collation ID, any string expression in the **ORDER BY** clause is treated as if the **SORTKEY** function has been invoked.
- The **SORTKEY** function uses the International Components for Unicode (ICU) library, instead of the Sybase Unicode Infrastructure Library (Unilib®). Sort key values created using a version of Sybase IQ earlier than 15.0 do not contain the same values created using version 15.0 and later. Regenerate any sort key values in your database that were generated using a version of Sybase IQ earlier than 15.0.
- The **CREATE DATABASE** parameter **COLLATION** supports specification of a collation for a database.

  The collation of the database must match the collation used by the operating system. In the Sybase Central Create Database wizard, the default character set is ISO_BINENG; change this, if necessary, to match the character set used by the operating system.
- The **CP874toUTF8** utility converts data in the CP874 character set into UTF8 collation, a collation supported by Sybase IQ for the Thai language. The **CP874toUTF8** utility calls the ICU library to perform data conversion. You can also load data in the CP874 character set without converting the data to UTF8 using this utility. See the *Utility Guide > CP874toUTF8 Database Administration Utility*.

Sybase IQ no longer supports custom collations. If you are rebuilding a database with a custom collation, the collation is preserved if you rebuild in a single step. If you unload the database and then load the schema and data into a database that you create, then you must use one of the supplied collations.

For an introduction to the issues you may face when working in an environment that uses more than one character set, or when using languages other than English, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Localized versions of SQL Anywhere > SQL Anywhere international features*.

For more information on changes to database collations and a list of collations deprecated in Sybase IQ 15.0, see *New Features in Sybase IQ 15.0*. Use the **iqunload** utility to migrate to Sybase IQ 15.3 from an existing 12.6 or 12.7 database that was created with a deprecated collation. For details about **iqunload**, see the *Installation and Configuration Guide*.

See also

- *How to Start the iqdemo Database* on page 44
- *Server Names* on page 25
- *Ways to Start Database Servers* on page 19
- *Creating an ODBC Data Source from the Command Line* on page 65
- *Starting a Server from Interactive SQL* on page 33

Default Collation

If you use the default actions when creating a Sybase IQ database, your database has the collation ISO_BINENG.

ISO_BINENG collation provides optimal performance for IQ databases, but not necessarily the most natural sort order.

This differs from SQL Anywhere, which infers the default collation for new databases from the character set in use by the operating system on which the database is created.

If you cannot set up your system in this default manner, decide which collation to use in your database, and whether to use character set translation to ensure that data is exchanged consistently between the pieces of your database system.

Understanding Character Sets in Software

Each piece of software works with a character set, which is a set of symbols, including letters, digits, spaces, and other symbols.

For general information about software issues related to international languages and character sets, including single-byte and multibyte character sets, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Understanding character sets*.

Code pages in Windows

Many languages have few enough characters to be represented in a single-byte character set. In such a character set, each character is represented by a single *byte*: a two-digit hexadecimal number.

At most, 256 characters can be represented in a single byte. No single-byte character set can hold all of the characters used internationally, including accented characters. This problem was addressed by the development of a set of *code pages*, each of which describes a set of characters appropriate for one or more national languages. For example, code page 869 contains the Greek character set, and code page 850 contains an international character set suitable for representing many characters in a variety of languages.

For information on ANSI and OEM code pages in Windows, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Understanding character sets > ANSI and OEM code pages in Windows*.

For a list of supported code pages, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Character set and collation reference information > Supported and alternate collations*.

How the Collation Sequence Sorts Characters

The database collation sequence includes alphabetic ordering of letters, and extends it to include all characters in the character set, including digits and space characters.

More than one character can be associated with each sort position. This is useful if you wish, for example, to treat an accented character the same as the character without an accent.

Two characters with the same sort position are considered identical in all ways by the database. Therefore, if a collation assigned the characters *a* and *e* to the same sort position, then a query with the following search condition:

```
WHERE col1 = 'want'
```

is satisfied by a row for which **col1** contains the entry **went**.

At each sort position, lower- and uppercase forms of a character can be indicated. For case-sensitive databases (the default for Sybase IQ databases), the lower- and uppercase characters are not treated as equivalent. For case-insensitive databases, the lower- and uppercase versions of the character are considered equivalent.

Note: Any code that selects a default collation for a German system should select 1252LATIN1, not 1252DEU. 1252DEU differentiates between characters with and without an umlaut, while 1252LATIN1 does not. 1252LATIN1 considers Muller and Müller equal, but 1252DEU does not consider them equal. Because 1252DEU views characters with umlauts as separate characters, it has the following alphabetic ordering: ob, öa.

First-Byte Collation Orderings for Multibyte Character Sets

A sorting order for characters in a multibyte character set can be specified only for the first byte.

Characters that have the same first byte are sorted according to the hexadecimal value of the following bytes.

Understanding Locales

Both the database server and the client library recognize their language and character set environment using a locale definition.

See *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Understanding locales*.

Understanding Collations

A collation describes how to sort and compare characters from a particular character set or encoding.

For information about the supplied collations, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database >International languages and character sets > Understanding collations*.

For suggestions as to which collations to use under certain circumstances, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database >International languages and character sets > Character set and collation reference information*.

For compatibility collations that can be used with the SORTKEY and COMPARE functions, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database >International languages and character sets > Character set and collation reference information > Supported and alternate collations*.

For a list of character sets, code pages, encodings and collations recommended for use with Windows and Unix platforms, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database >International languages and character sets > Character set and collation reference information > Recommended character sets and collations*.

Displaying Collations

Obtain the currently used collation.

Type the following command at the command prompt:

```
SELECT * FROM SYS.SYSCOLLATION
```

ANSI and OEM Code Pages

Sybase IQ collations are based on code pages that are designated as either ANSI or OEM.

In most cases, use of an ANSI code page is recommended. If you use OEM, choose a code page that matches the OEM code pages on your users' client machines.

You should not use a separate translation driver under any circumstance. Translation drivers interfere with the server's character set translation. Using a separate translation driver will likely result in data corruption.

For Interactive SQL and Sybase Central, the iAnywhere JDBC driver handles character set translation.

ANSI ISO_1 Collation

ISO_1 is provided for compatibility with the Adaptive Server Enterprise default ISO_1 collation.

The differences with Adaptive Server Enterprise are:

- ß, the lowercase letter sharp s (\xDF), sorts after lowercase **s** in Sybase IQ and SQL Anywhere, but after **ss** in Adaptive Server Enterprise.
- The ligatures corresponding to **Æ** and **æ** (\xC6 and \xE6) sort after **A** and **a** respectively in Sybase IQ and SQL Anywhere (**A**, **a**, **æ**, **Æ**). In Adaptive Server Enterprise the sort order is **A**, **a**, **Æ**, **æ**.

ANSI 1252LATIN1 Collation

This collation includes the euro currency symbol and several other characters (Z-with-caron and z-with-caron).

For single-byte Windows operating systems, this is the recommended collation in most cases. This collation is recommended for Windows users using English or Western European languages.

ANSI ISO1LATIN1 Collation

This collation is the same as ISO_1, but with sorting for values in the range A0-BF.

For compatibility with Adaptive Server Enterprise, the ISO_1 collation has no characters for 0xA0-0xBF. However, the ISO Latin 1 character set on which it is based does have characters in these positions. The ISO1LATIN1 collation reflects the characters in these positions.

If you are not concerned with Adaptive Server Enterprise compatibility, ISO1LATIN1 is generally recommended instead of ISO_1.

ISO1LATIN1 is recommended for UNIX users using English or Western European languages, if you are willing to sacrifice some of the optimal performance of the default collation, ISO_BINENG.

ANSI ISO9LATIN1 Collation

This collation is the same as ISO1LATIN1, but includes the euro currency symbol and the other new characters included in the 1252 LATIN1 collation.

If your machine uses the ISO Latin 9 character set, and you are willing to sacrifice some of the optimal performance of ISO_BINENG, use this collation.

Multibyte Collations

Sybase IQ provides collations using several multibyte character sets.

Sybase IQ supports variable-width character sets. In these sets, some characters are represented by one byte, and some by more than one, to a maximum of four bytes. The value of the first byte in any character indicates the number of bytes used for that character, and also indicates whether the character is a space character, a digit, or an alphabetic (alpha) character.

For the UTF8 collation, UTF-8 characters are represented by one to four bytes. For other multibyte collations, one or two bytes are used. For all provided multibyte collations, characters comprising two or more bytes are considered to be “alphabetic”, such that they can be used in identifiers without requiring double quotes.

Sybase IQ does not support 16-bit or 32-bit character sets such as UTF-16 or UTF-32.

All client libraries other than embedded SQL are Unicode-enabled, using the UTF-16 encoding. Translation occurs between the client and the server.

Japanese Language Support

Sybase recommends using collation 932JPN for Japanese Windows applications.

Collation 932JPN supports loading 32-bit multibyte characters that cannot be loaded into SJIS or SJIS2. SJIS and SJIS2 are older collations. SJIS is available as an alternate collation. SJIS2 is no longer supported. For Unix applications, use EUC_JAPAN.

Thai Language Support

Sybase IQ provides the **CP874toUTF8** utility to convert data files in CP874 format into UTF8, a collation supported by Sybase IQ for the Thai language.

For syntax, see the *Utility Guide*. You can also load data in the CP874 character set without converting it to UTF8 using this utility.

The SORTKEY() function returns values in the sort order **thaidict** (Thai dictionary), the Thai character set in UTF8 form. For syntax see Reference: Building Blocks, Tables, and Procedures. The following statements generate the same result:

```
SELECT c1, SORTKEY(c1) from T1 where rid=3
SELECT c1, SORTKEY(c1, ‘thaidict’) from T1 where rid=3)
SELECT
‘\340\270\201\340\271\207’,SORTKEY(‘\340\279\201\340\271\207’) from
T1 where rid=3
```

Understanding Character Set Translation

Sybase IQ can carry out character set translation among character sets that represent the same characters, but at different positions in the character set or code page.

There needs to be a degree of compatibility between the character sets for this character set translation to be possible. For example, character set translation is possible between EUC-JIS and cp932 character sets, but not between EUC-JIS and cp1252.

These topics describe how Sybase IQ carries out character set translation. This information is provided for advanced users, such as those who may be deploying applications or databases in a multicharacter set environment.

Character Translation for Database Messages

Error and other messages from the database software are held in a language resource library. Localized versions of this library are provided with localized versions of Sybase IQ.

Client application users may see messages from the database as well as data from the database. Some database messages, which are strings from the language library, may include placeholders that are filled by characters from the database. For example, if you execute a query with a column that does not exist, the returned error messages is:

```
Column column-name not found
```

where *column-name* is filled in from the database.

To present these kinds of information to the client application in a consistent manner, even if the database is in a different character set from the language library, the database server automatically translates the characters of the messages so that they match the character set used in the database collation.

Messages are always translated into the database collation character set, regardless of whether character set conversion is turned on or off.

A further character set translation is carried out if character set translation is turned on (the default) for the database server, and if the client character set is different from that used in the database collation.

Using Character Translation for Database Messages

Perform checks before reading database messages.

1. Ensure that the collation for your database is compatible with the character set used on your computer, and with the character set used in the Sybase IQ language resource library. The language resource library differs among different localized versions of Sybase IQ.
2. Check that the characters of interest to you exist in each character set.

Connection Strings and Character Sets

Connection strings present a special case for character set translation.

The connection string is parsed by the client library, in order to locate or start a database server. This parsing is done with no knowledge of the server character set or language.

The interface library parses the connection string as follows:

- It is broken down into its *keyword= value* components. This can be done independently of character set, as long as you do not use the curly braces {} around CommLinks parameters. Instead, use the recommended parentheses (). Curly braces are valid follow bytes (bytes other than the first byte) in some multibyte character sets.
- The server is located. The server name is interpreted according to the character set of the client machine. In the case of Windows operating systems, the ANSI character set is used. Extended characters can be used unless they cause character set conversion issues between client and server machine.

  For maximum compatibility among different machines, you should use server names built from alphabetic ASCII characters 1 to 127 (or 33 to 126) and the underscore, using no punctuation characters. Server names are truncated at 40 characters.
- The DatabaseName (DBN) or DatabaseFile (DBF) parameter is interpreted in the database server character set.
- Once the database is located, the remaining connection parameters are interpreted according to its character set.

See also

- *Connection and Communication Parameters* on page 83
- *Simple Connection Examples* on page 53

Avoiding Character-Set Translation

There is a performance cost associated with character set translation.

If you can set up an environment such that no character set translation is required, then you do not have to pay this cost, and your setup is simpler to maintain.

If you work with a single-byte character set and are concerned only with seven-bit ASCII characters (values 1 through 127), then you do not need character set translation. Even if the code pages are different in the database and on the client operating system, they are compatible over this range of characters. Many English-language installations will meet these requirements. In this version character set translation is turned on by default.

If you do require use of extended characters, there are other steps you may be able to take:

- If the code page on your client machine operating system matches that used in the database, no character set translation is needed for data in the database.

- If you can use a version of Sybase IQ built for your language, and if you use the code page on your operating system, no character set translation is needed for database messages. The character set used in the Sybase IQ message strings is as follows:

Language	Character set
English	cp1252
French	cp1252
German	cp1252
Japanese	cp932 (Shift-JIS)

Also, recall that client/server character set translation takes place by default. Character set translation is disabled if you specify **CharSet=none** in the connection string.

Configuring Your Character Set Environment

Set up your computing environment so that character set issues are handled properly.

1. Determine the default locale of each computing platform in your environment. The default locale is the character set and language of each computer. On Windows operating systems, the character set is the ANSI code page.
2. Decide whether the locale settings are appropriate for your environment.
3. If the default settings are inappropriate, decide on a character set, language, and database collation that match your data and avoid character set translation.
4. Set locales on each of the machines in the environment to these values.
5. Create your database using the default collation. If the default collation does not match your needs, create a database using a named collation.
6. When choosing the collation for your database, consider the following:
   a) Choose a collation that uses a character set and sort order appropriate for the data in the database. It is often the case that there are several alternative collations that meet this requirement, including some that are OEM collations and some that are ANSI collations.
   b) Choose a collation that avoids the need for character set translation. There is a performance cost, as well as extra complexity in system configuration, when you use character set translation. You can avoid character set translation by using a collation sequence in the database that matches the character set in use on your client machine operating system. In the case of Windows operating systems on the client machine, choose the ANSI character set. Character set translation is enabled by default for Sybase IQ database servers that are version 15.3 or higher. You can turn off character set translation using the **CharSet=none** on the command line.

For more information on character sets, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and* *character sets > Character set and collation reference information > Supported character sets*.

Locale Information

Use system functions to determine locale information.

To see how to use system functions to return locale information about the client connection, database, and database server, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > International language and character set tasks > Determining locale information*.

See also

- *CharSet connection parameter [CS]* on page 88
- *Language Connection Parameter [LANG]* on page 102
- *Setting a Locale* on page 400

Setting a Locale

You can use the default locale on your operating system, or explicitly set a locale for use by the Sybase IQ components on your machine.
Set either or both of the IQLANG and SACHARSET environment variables:

```
SACHARSET=charset;IQLANG=language_code
```

where *charset* is a valid character set label and *language_code* is a language code from the list of language label values in *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > Understanding locales > Understanding the locale language*.

Sybase IQ loads the localization information when it executes the **INSERT...LOCATION** statement.

See also

- *How Connection Parameters Work* on page 47
- *Locale Information* on page 400
- *CharSet connection parameter [CS]* on page 88
- *Language Connection Parameter [LANG]* on page 102

Setting the Locale for an INSERT...LOCATION Statement

When the database uses a non-default locale for your platform, you must set an environment variable on the local client in order for Sybase IQ to load the correct information for language, collation sequence, character set, and date/time format.

When determining the locale name, Sybase IQ first checks for the value of the LC_ALL environment variable. If LC_ALL is not set, Sybase IQ uses the value of the LANG environment variable. If neither variable is set, Sybase IQ uses the “default” entry in the locales file.

1. Open the `$SYBASE/locales/locales.dat` file in a text editor.

   For example:

   ```
   locale = default, us_english, roman8
   locale = C, us_english, roman8
   locale = american, us_english, roman8
   locale = english.iso88591, us_english, iso_1
   ```

2. Set the LC_ALL or LANG environment variable to the correct value. If on the platform in step 1, your database’s collation is iso_1 and you are using English, then you need to set the value of the environment variable LC_ALL or LANG to “american.iso88591”. Otherwise, Sybase IQ will use the locale name “default” which has collation “roman8”.

   For example, in the sh or ksh shells:

   ```
   LC_ALL= american.iso88591;export LC_ALL
   ```

   In the csh or tsch shell:

   ```
   setenv LC_ALL american.iso88591
   ```

Creating a Database with a Named Collation

The default collation for an IQ database is always ISO_BINENG. You can specify a different collation for each database when you create it.

For information on creating a database with a named collation, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database > International languages and character sets > International language and character set tasks > Creating a database with a named collation*.

Disabling Character Set Translation on a Database Server

You can turn character set conversion on and off explicitly on the database server command line.

Character set translation takes place if the client and server locales are different. Character set translation is enabled by default in Sybase IQ.

Connect using the `CharSet=none` parameter on the connection string. For example:

```
CharSet=none
```

Changing a Database From One Collation to Another

Changing a database to another collation requires a rebuild of the database. Collations are chosen when databases are created, and cannot be changed.

For information on changing the collation by rebuilding the database, see *SQL Anywhere 11.0.1 > SQL Anywhere Server – Database Administration > Configuring Your Database >International languages and character sets >International language and character set tasks > Changing a database from one collation to another*.

Compatibility Issues

In versions earlier than 12.0, Sybase IQ always used the ASCII sort order, which sorts uppercase characters before lowercase.

In 12.4.2 and later versions, by default IQ databases sort data in the same way as pre-version 12 Sybase IQ. The default applies these **CREATE DATABASE** options:

```
CREATE DATABASE dbname
COLLATION 'ISO_BINENG'
BLANK PADDING ON
CASE RESPECT
```

With these options, uppercase characters precede all lowercase characters in the collation sequence. For example, 'XYZ' sorts before 'abc' with these options, just as it did in older versions of Sybase IQ.

Performance Issues

Performance for character data is better with a binary character set and collation sequence than with a nonbinary character set and collation sequence.

To maximize performance, create a database with these default option settings:

```
CREATE DATABASE dbname
COLLATION 'ISO_BINENG'
CASE RESPECT
```

These options result in a binary character set and collation sequence. All other settings for these two options form a nonbinary character set and collation sequence.

The disadvantage of these settings is that uppercase characters are always sorted before lowercase ones. For example, “BANANA” sorts before “apple.” If you prefer a more natural sort order, but still need a case sensitive database, and if you are willing to sacrifice some degree of performance, use the collation ISO_1 instead of the default, ISO_BINENG.

Note: For details about password case-sensitivity, see Reference: Building Blocks, Tables, and Procedures > Compatibility with Other Sybase Databases > Data definition language > Case-sensitivity.

Data Backup, Recovery, and Archiving

To protect your data, schedule and perform regular backups. You can also use read-only hardware to archive non-modifiable data for easy access.

Data Protection

Sybase IQ provides a full set of features that protect you from two types of computer failure, and from database inconsistency.

- A *system failure* occurs when the computer or operating system goes down while there are partially completed transactions. This could occur when the computer is inappropriately turned off or rebooted, when another application causes the operating system to crash, or because of a power failure.
- A *media failure* occurs when the database file, the file system, or the device storing the database file, becomes unusable.

After a system failure, Sybase IQ can usually recover automatically, so that you may not need to restore your database.

After media failure, or if for any reason the data in your database is inconsistent, you must restore your database. To protect your data in all of these situations, make regular backups of your databases. In particular, you should back up your database each time you finish inserting any large quantities of new data into the database.

When failures occur, the recovery mechanism treats transactions properly, as atomic units of work: any incomplete transaction is rolled back and any committed transaction is preserved. This ensures that even in the event of failure, the data in your database remains in a consistent state.

See also

- *Operating System Session Shutdown* on page 42
- *Resource Issues Running sp_iqcheckdb* on page 461
- *System Recovery* on page 385
- *System Recovery and Database Repair* on page 453
- *Validating Your Database* on page 426

How to Back Up Databases

Use the **BACKUP** command to back up your IQ database.

Backup includes both the Sybase IQ data (the IQ store) and the underlying SQL Anywhere database (the catalog store).

Backup runs concurrently with read and write operations in the database. By contrast, during a restore no other operations are allowed on that database.

You must be connected to a database to back it up. The **BACKUP** command has no way to specify another database.

For information about backing up multiplex databases, see *Using Sybase IQ Multiplex > Backup and Restore Operations in Multiplex Environments*.

Types of Data Stores

Sybase IQ data stores consist of one or more files.

They can contain both user data and internal database structures used for startup, recovery, backup, and transaction management. Typically, an IQ database has the following stores:

- *db-name.db* is the catalog dbspace containing the system tables and stored procedures describing the database and any standard SQL Anywhere database objects you add. It is known as the catalog store, and has the dbspace-name `SYSTEM`. You can create additional dbspaces in the catalog store.
- *db-name.iq* is the main data dbspace containing the IQ table data and indexes. It is known as the IQ store, and has the dbspace-name `IQ_SYSTEM_MAIN`. The dbfile name matches dbspace-name, `IQ_SYSTEM_MAIN`. You can create multiple dbspaces in the IQ store, and each dbspace can hold multiple dbfiles, including `IQ_SYSTEM_MAIN`.
- *db-name.iqtmp* is the initial temporary dbspace containing the temporary tables generated by certain queries. It is known as the IQ temporary store and has the dbspace-name `IQ_SYSTEM_TEMP`. You can add dbfiles to the IQ temporary store.

Any of these stores, and the log files, are possible areas of failure.

Types of Backups

There are four ways to back up Sybase IQ data.

- Database backup
- Operating system-level backup
- Virtual backup
- Archive backup (for log files)

Types of Database Backups

Sybase IQ provides four types of database backups:

- *Full backup* makes a complete copy of the database.
- *Virtual backup* copies all of the database except the table data from the IQ store.
- *Incremental backup* copies all transactions since the last backup of any type.
- *Incremental-since-full backup* copies all transactions since the last full backup.

All these backup types fully back up the catalog store. In most cases, the catalog store is much smaller than the IQ store. If the catalog store is larger than (or nearly as large as) the IQ store, however, incremental backups of IQ are bigger than you may want or expect.

Incremental virtual backup is supported using the **BACKUP** statement.

Temporary store data is not backed up. However, the metadata and any other information needed to recreate the temporary store structure is backed up.

Backing Up the IQ Store and Catalog Store

Read the rest of the backup topics for complete details before you perform a backup.

This procedure summarizes backup steps.

1. Connect to the server using an account with DBA privileges. For a multiplex database, you must connect to the coordinator.
2. Run the **BACKUP** command. For complete syntax, see *Reference: Statements and Options*.

   It backs up the following files:
   - The catalog store (SYSTEM dbspace file), typically named *dbname*.db
   - All dbspace files of the IQ store
3. Make a copy of the `params.cfg` file for each server. **BACKUP** does not back it up.
4.

   Save the lengths of the following files:
   - All dbspace files on the coordinator
   - IQ temporary store

Data in Backups

BACKUP *backs up committed data only.*

Backups begin with a commit and an automatic checkpoint. At this point, the backup program determines what data will be backed up. It backs up the current snapshot version of your database as of the time of this checkpoint. *Any data that is not yet committed when this checkpoint occurs is not included in the backup.*

A second automatic checkpoint occurs at the end of backup. Any data that is committed while the backup is in progress is included in any subsequent backups.

Sybase IQ backs up only those recoverable database blocks actually in use at the time of backup. Free blocks are not backed up.

Sybase IQ backs up the database files and the catalog information that pertains to the IQ database to which you are connected. *It does not back up the transaction log file.* It does not use the transaction log to restore the database.

If for any reason all the commands in the transaction do not process properly, or your database is missing files, the backup fails.

The Transaction Log In Backup, Restore, and Recovery

Sybase IQ uses the transaction log file during recovery from a system failure.

It does not use the transaction log to restore an IQ database, to recover committed IQ transactions, or to restore the catalog store for a Sybase IQ database. For a full restore, the transaction log must not exist. You must delete this file before starting a full restore.

Note: SQL Anywhere databases use the transaction log and other logs differently. If you are recovering a SQL Anywhere database, you need its transaction log file, and **BACKUP** retains it for you. See *SQL Anywhere Server – SQL Usage* for details. Also, if you have data (other than the system tables) in your catalog store, transactions for that data can only be recovered if they were written to disk before a failure.

Making a Live Backup of a Transaction Log

You can make a live backup of the transaction log using the **dbbackup** utility with the **-l** option.

1. Set up a secondary machine from which you can run the database if the online machine fails. (Install and configure Sybase IQ on the secondary machine.)
2. Periodically, make a full backup to the secondary machine.
3.

   Run a live backup of the transaction log to the secondary machine.

   ```
   dbbackup -l path\filename.log -c "connection_string"
   ```

   You should normally run the **dbbackup** utility from the secondary machine. If the primary machine becomes unusable, you can restart your database using the secondary machine. The database file and the transaction log hold the information needed to restart.

Distribution of Backup Data

BACKUP always makes a full backup of the catalog store on the first archive device, and then backs up the data from the IQ store in parallel across all of the devices you specify.

Blocks are not evenly distributed across archive media. You may have more on one device than others, depending on the processing speed of individual threads.

Note: The distribution of backup data is important because sets of files must be restored in the order in which they were backed up.

See also

- *Getting Information about Backups and Restores* on page 441
- *Restoring in the Correct Order* on page 434

Ensure that your Database is Consistent

Although **BACKUP** does check that all necessary files are present before backing up your database, it does not check internal consistency.

For a more thorough check, you can run the stored procedure **sp_iqcheckdb** before making a backup.

See also

- *Validating the Database After You Restore* on page 437
- *Validating Your Database* on page 426

Select Archive Devices

You can back up any IQ database onto magnetic tape or disk, including WORM devices.

Sybase IQ supports backup and restore using multiple tape drives at near device speeds, or to multiple disks if disk striping is in use. Specify the backup device name in the *archive_device* parameter of the **BACKUP** command.

See also

- *Archiving Data with Read-Only Hardware* on page 448

Disk Backup Requirements

Disk backups must go to a file system; raw disk is not supported as a backup medium. All disks on a redundant array of independent devices (RAID) device are treated as a single device.

Tape Backup Requirements

If you regularly back up large databases, you should use DLT drives, if they are supported for your platform. In any case, Sybase recommends that you use multiple tape drives.

Sybase IQ **BACKUP** can support the following tape drives:

- Digital Linear Tape (DLT) on UNIX systems
- 4 mm Digital Data Storage (DDS)
- 8 mm

Sybase IQ also allows Stacker drives with multiple tapes.

Sybase IQ **BACKUP** does *not* support jukeboxes or robotic loaders. If you need them, use a third party media manager.

Sybase IQ **BACKUP** does *not* support fixed-length tape devices on UNIX systems, like Quarter Inch Cartridge (QIC) drives.

Platform-specific Backup Requirements

Be aware of backup requirements for AIX and IBM Linux.

Be aware of the following platform-specific backup requirements:

- Tape devices on AIX systems can be configured for either fixed- or variable-length block mode. See the *Installation and Configuration Guide* for information on how to show and change the block mode. Sybase IQ **BACKUP** does not support fixed-length block mode.
- On IBM Linux on POWER, to back up an IQ database to SCSI tape, you must set the block size of the device to accept variable-length data transfer. Before performing any IQ backups, set the SCSI tape device's default block size. Log in as superuser and run the Linux shell command **mt**, as follows:

  ```
  mt -f /dev/st0 defblksize 0
  ```

Limits on the Number of Backup Devices

When using the **BACKUP** command of Sybase IQ, users can parallelize the operation to multiple devices by specifying multiple **TO** clauses. The backup statement has an upper limit of 36 on the device numbers. If the upper limit is exceeded, the statement produces an error.

This limit affects all versions of Sybase Risk Analytics Platform and Sybase RAP - The Trading Edition™.

Corrective Actions for Future Backups

Users must create future backups using 36 or fewer **TO** clauses in a **BACKUP** command, a limit that will be enforced in future versions of Sybase IQ.

Keep backup commands small. Do not to go to extremes with the number of devices, as this will cause I/O and hardware contention with more devices. As a practical guideline, use roughly 1 device per core on the machine to saturate CPU usage. Use up to 2 devices per core on faster systems.

Preparing for Backup

In order to run **BACKUP**, you must meet the requirements described in the sections that follow.

Obtaining DBA Privileges

You need DBA privileges on a database to run **BACKUP** or **RESTORE**.

You must either log on as the DBA user, or be granted DBA authority by the DBA.

Rewind Tapes

Sybase IQ does not rewind tapes before using them.

You must ensure the tapes used for backup or restore operations are at the correct starting point before putting them in the tape device.

Tapes are rewound after the backup if you are using a rewinding device. If your tape device automatically rewinds tapes, take care that you do not overwrite any information on the tape.

Retain Old Disk Backups

BACKUP overwrites existing disk files of the same name.

If you need to retain a backup, when you create a new backup either use different file or path names for the archive devices, or move the old backup to another location before starting the backup.

Two Ways to Run BACKUP

You can run **BACKUP** in two ways.

- **Attended –** In attended mode, **BACKUP** assumes that an operator is present, and prompts you to mount the archive media when necessary. With this method, you must run **BACKUP** interactively from the command line.
- **Unattended –** In unattended mode, **BACKUP** assumes that no operator is present, and does not issue prompts. Instead, you must make appropriate estimates of the space required, and set up your devices accordingly. Any error is considered fatal.

In some cases, you can use third party software to create backups. Such products can be particularly useful for unattended backups.

Note: You can run **BACKUP** from a batch script or procedure, as well as from Interactive SQL. You can also automate backups using an event handler. See *System Administration Guide: Volume 2*.

See also

- *Unattended Backup* on page 413
- *Estimate Media Capacity* on page 411

Estimate Media Capacity

Before you do a backup, be sure that your archive media has sufficient space.

When you estimate available space on disk or tape, keep in mind these rules:

- You need enough room for a full backup of the catalog store, as well as the full or incremental backup of the IQ store. If your catalog store holds SQL Anywhere data in addition to the Sybase IQ system tables, you need room to back up this data as well.
- You do not need to include space for the transaction log, as this log is not backed up.
- For tape backups, the first tape set you specify must be able to hold the full backup of the catalog store, including any non-IQ data in the catalog store. (A tape set consists of one or more backup tapes produced on a given archive device.)
- For stacker devices that hold multiple tape drives, all tapes for a given device must be the same size.

Sybase recommends that you always start a new tape for every backup.

Before starting a backup to disk, Sybase IQ first tests whether there is enough disk file space for the backup. For an operator-attended backup to disk, if there is not enough space, **BACKUP** prompts you to move some files from the disk before it writes any data. The backup does not start until you provide more disk space.

Likewise, if you run out of space during an attended disk backup, **BACKUP** closes all open backup files and waits until it detects that you have cleared some space. Then it restarts with new backup files. You can also stop the backup if you prefer.

By default, you must provide at least 8KB of free disk space before the backup resumes.

Unattended backup cannot prompt you to provide more space. Unless enough space is available, unattended backup fails. **BACKUP** treats size estimates differently for unattended backups.

For an operator-attended backup to tape, **BACKUP** simply begins the backup. If it runs out of room, you must mount additional tapes.

Running Backups

Use the BACKUP statement to run backups.

See also

- *How Transaction Information Aids Recovery* on page 386

Concurrency and Backups

You can run backups concurrently with most other database operations.

The exception are:

- No metadata changes can occur while the catalog store is backed up.
- No commands that issue checkpoints or DBCC can be run during backup.

Be aware, however, that transactions that have not committed when you start a backup are not backed up. If a system or media failure occurs during backup, you cannot restore uncommitted transactions.

Once a backup is started, you cannot execute a **CHECKPOINT** command.

Specify Operator Presence

ATTENDED ON or **OFF** controls whether or not human intervention is expected when new tapes or disk files are needed.

The default is **ON**.

For unattended backups to disk, **BACKUP** does not prompt you to add more disk space. If you run out of space, an error occurs and **BACKUP** halts.

For unattended backups to tape, **BACKUP** does not prompt for a new tape to be loaded. The **SIZE** and **STACKER** options determine what happens if you run out of space.

Unattended Backup

With the **ATTENDED OFF** option, you can specify that no operator will be present during a backup.

Sybase IQ supports two unattended backup features:

- The operator does not need to respond to prompts during the backup.
- The archive devices can be stacker drives, which automatically load a set of tapes into a single drive. You can use stacker drives for both attended and unattended backups.

Unattended backup tries to detect all possible reasons for a backup failing except tape media failure, and report any potential errors before attempting the backup, such as available space on disk or tape, and consistent size and block factor.

For unattended backup to disk, Sybase IQ first tests whether there is enough free disk space for the backup, However, it does not preallocate the backup files to reserve the space. If another user writes to that disk and as a result there is not enough room for the backup, the backup fails when disk space runs out.

For backup to tape, you must estimate how much data each tape will hold, and specify that number of kilobytes in the **TO** *archive_device* parameter of the **BACKUP** command. The backup program checks information stored internally to see how much room it needs to back up your database. If it determines that there is enough room on the tape, the backup proceeds. However, if you overestimate the amount of space available on the tape(s) and the backup runs out of space, the backup fails at that point.

If you omit the **SIZE** parameter for an unattended backup, the entire backup must fit on one tape.

If you are using a third-party backup product, the vendor information string needs to convey any information needed for the backup, such as the specification of devices, size of files, and stacker drives. See your vendor's documentation for details.

Note: Sybase IQ does not permit unattended restore.

Specify the Type of Backup

FULL | INCREMENTAL | INCREMENTAL SINCE FULL specifies the type of backup.

Choose one:

- **FULL** causes a full backup of both the catalog store and the IQ store. **FULL** is the default action.

  For a virtual backup, you can use the **VIRTUAL DECOUPLED | VIRTUAL ENCAPSUATED** options of the **BACKUP** statement.
- **INCREMENTAL** makes a full backup of the catalog store, and then backs up all changes to the IQ store since the last IQ backup of any type.

- **INCREMENTAL SINCE FULL** makes a full backup of the catalog store, and then backs up all changes to the IQ store since the last full IQ backup.

INCREMENTAL and **INCREMENTAL SINCE FULL** virtual backups are supported using the **VIRTUAL DECOUPLED** and **VIRTUAL ENCAPSULATED** options of the **BACKUP** statement.

You may restrict full, incremental-since-full, or incremental backup to the set of read-write files in the databases using the READWRITE FILES ONLY keywords. The read-write dbspaces or files that are backed up must belong to the IQ main store. The backed up files are selected when the backup command checks the read-write status in the catalog.

An IQ backup may back up a set of read-only dbspaces and/or read-only files. The read-only dbspaces or files must belong to the IQ main store. The backed up files are user selected.

See also

- *Scheduling Routine Backups* on page 445

Specifying Virtual Backup

The **VIRTUAL DECOUPLED | VIRTUAL ENCAPSUATED** '*shell-command* options specify the type of virtual backup.

The *shell-command* variable of the `VIRTUAL ENCAPSULATED` parameter allows shell commands to execute a system-level backup as part of the backup operation.

See also

- *Virtual Backups* on page 421

Specifying Archive Devices

The **TO** *archive_device* clause indicates the destination disk file(s) or system tape drive(s) for the backup and controls the number of archive devices.

Backup File Names for Backup to Disk

BACKUP always assigns file names to disk backup files by appending a suffix to the *archive_device* name you specify.

The suffix consists of *"."* followed by a number that increases by one for each new file. For example, if you specify `/iqback/mondayinc` as the *archive_device*, the backup files are `/iqback/mondayinc.1`, `/iqback/mondayinc.2`, and so on. This convention allows you to store as large a backup as you need, while allowing you control over the file size; see the **SIZE** option for details. Your file system must support long file names to accommodate this convention.

You must make sure that the directory names you specify for the *archive_device* exist. **BACKUP** does not create missing directories. If you try to start a backup in a directory that does not exist, the backup fails.

You should avoid using relative path names to specify the location of disk files. **BACKUP** interprets the path name as relative to the location where the server was started, which you may not be able to identify with certainty when you do a backup. Also, if there is data in other directories along the path, you may not have enough room for the backup.

Positioning Tape Devices

BACKUP does not position tapes for you. You must position the tape appropriately before starting your backup, and be sure that you do not overwrite any of the backup if you use a rewinding tape device. For these reasons, Sybase recommends you use a non-rewinding tape device. See the operating system documentation for your platform for appropriate naming conventions.

Specifying Tape Devices on UNIX

Here are examples of how you specify non-rewinding tape devices on UNIX platforms:

- On Sun Solaris platforms, insert the letter n for “no rewind” after the device name, for example, '`/dev/rmt/0n`'.
- On IBM AIX platforms, use a decimal point followed by a number that specifies the appropriate compression with rewind setting, for example, '`/dev/rmt0.1`'.
- On HP-UX platforms, use '0m' to specify the default tape mechanism and 'n' for “no rewind,” for example, '`/dev/rmt/0mn`'.

Warning! If you misspell a tape device name and write a name that is not a valid tape device on your system, **BACKUP** assumes it is a disk file.

Specifying Tape Devices on Windows

Windows systems do not specify rewind or no rewind devices and only support fixed-length I/O operations to tape devices. Sybase IQ requires variable-length devices. It does additional processing to accommodate fixed-length tape I/O on Windows systems.

While Windows supports tape partitioning, Sybase IQ does not use it, so do not use another application to format tapes for Sybase IQ backup or restore. On Windows, the first tape device is '`\\.\tape0`', the second is '`\\.\tape1`', and so on.

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

Specify the Size of Tape Backups

The **SIZE** option of the **TO** clause identifies the maximum size of the backed up data on that stripe, in KB.

If you use the Sybase-provided backup (as opposed to a third party backup product), you should specify **SIZE** for *unattended* tape backups on platforms that do not reliably detect the end-of-tape marker. Note that the value of **SIZE** is per output device. No volume used on the corresponding device can be shorter than this value. Although IQ does not require you to specify **SIZE** for an *attended* tape backup, it is always best to supply an accurate size estimate.

During backup, if any tape runs out of space and you have not specified **SIZE**, you get an error. If any tape runs out of space before the specified size, you do not get an error immediately; instead, here is what happens:

- For attended backups with **SIZE** and **STACKER** specified, Backup tries to open the next tape.
- For attended backups with **SIZE** specified but not **STACKER**, Backup asks you to put in a new tape.
- For unattended backups with **SIZE** and **STACKER** specified, Backup tries to open the next tape. If there are no volumes available, or if you did not specify **STACKER**, you get an error.

Any additional tapes do not contain the header information needed for a restore, so you must be careful to mount tapes in order during the restore or your database could become inconsistent.

On Windows, there are special requirements for the **SIZE** option on tape devices:

- The value of **SIZE** must be a multiple of 64. Other values are rounded down to a multiple of 64.
- If you do not specify **SIZE** explicitly, it is automatically set to 1.5GB.

Specify the Size of Disk Backups

The **SIZE** option of the **TO** clause identifies the maximum size of the backed up data on that stripe, in KB. Note that the value of **SIZE** is per output device.

If you use the Sybase-provided backup, either attended or unattended, specify **SIZE** if any disk file you name as an *archive_device* is larger than the default of 2GB (UNIX) or 1.5GB (Windows).

During backup, when the amount of information written to a given *archive_device* reaches **SIZE**, backup closes the current file and creates another one of the same name with the next ascending number appended to the file name.

For example, if you specify one *archive_device*, a disk file called `janfull`, and you specify SIZE 200000 for a maximum 200MB file, but your backup requires 2GB, then **BACKUP** creates ten 200MB files: `janfull.1`, `janfull.2`,...`janfull.10`. You must ensure that your disk can accommodate this much data before performing the backup.

Specify Stacker Devices

The **STACKER** option of the **TO** clause indicates that you are backing up to an automatically loaded multitape stacker device, and specifies the number of tapes in that device. When **ATTENDED** is **ON** and **STACKER** is specified, **BACKUP** waits indefinitely for the next tape to be loaded. All tapes in a given stacker device must be the same size.

Specify Devices for Third Party Backups

Note: Do not specify **SIZE** or **STACKER** if you are using a third party backup product, as size information is conveyed in the *vendor_specific_information* string.

See also

- *Performing Backups with Non-Sybase Products* on page 420
- *Recording dbspace Names* on page 443
- *Restoring in the Correct Order* on page 434
- *The RESTORE Statement* on page 430
- *Verifying a Database Backup* on page 439

Other Backup Options

You may want to set a number of other **BACKUP** command options to customize your backup.

Specifying the Block Factor

BLOCK FACTOR specifies the number of IQ blocks to write to the archive device at one time.

It must be greater than 0, or **BACKUP** returns an error message. **BLOCK FACTOR** defaults to 25 on UNIX platforms. On Windows, the default **BLOCK FACTOR** is based on the block size of your database. For example, if the block size is 512 bytes, **BLOCK FACTOR** is 120 blocks. If the block size is 32KB, **BLOCK FACTOR** is 1 block.

This parameter also controls the amount of memory used for buffers during the backup, and has a direct impact on backup performance. The effects of the block factor are a function of disk subsystem speed, tape speed, and processor speed. Some systems have better backup performance with a smaller block factor, while others may have better backup performance with a larger one. See your platform operating system documentation for information about your platform’s optimal I/O size and block factor.

Specify Error Checking

CRC ON or **OFF** activates or deactivates 32-bit cyclical redundancy checking on a per block basis.

(**BACKUP** also uses whatever error detection is available in the hardware.) With **CRC ON**, the checksums computed on backup are verified during any subsequent **RESTORE** operation. The default is **CRC ON**.

Add Comments

WITH COMMENT specifies a string up to 32KB long as part of the header information for the backup archive.

If you omit this option, **BACKUP** enters a NULL. You can view the comment string by executing a **RESTORE DATABASE FROM CATALOG ONLY**, or by displaying the backup log, `backup.syb`, that Sybase IQ provides.

To back up an SQL Anywhere-only database, see *SQL Anywhere Server – SQL Reference* for additional **BACKUP** options.

Wait for Tape Devices

During backup and restore operations, if Sybase IQ cannot open the archive device (for example, when it needs the media loaded), the server waits for ten seconds and tries again.

The server continues these attempts indefinitely, until either the operation succeeds or is terminated with a Ctrl+C. A message is written to the server `.stderr` file. There is no console notification that the server cannot open the archive device.

Backup and Restore Using Read-Only Hardware

Sybase IQ supports read-only hardware for both backup and restore operations.

The following rules apply:

- Sybase IQ prevents writes to a read-only device during restore because the device may be frozen in read-only mode at the hardware level.
- Virtual backup will not back up or restore the header block of a read-only dbspace or any other block on a read-only dbspace. Since a read-only dbspace is guaranteed never to change, virtual backup and restore need only restore a read-only dbspace after media failure of the read-only dbspace.
- Non-virtual full backup will back up all dbspaces, regardless of mode.
- Non-virtual incremental backup will not back up read-only dbspaces that:
    - Were read-only at the time of the previous backup that the incremental backup depends on,
      and
    - Have not been altered since.

  The contents of such dbspaces are wholly contained by a previous depends-on backup. Read-only dbspaces that have been altered since the time of the depends-on backup are backed up.

Backup Examples

Example topics demonstrate backup options.

Example 1 — Full Backup

This example makes a full, attended backup of the database `iquser` to two tape devices on UNIX. Before running this backup you must position the tapes to the start of where the backup files will be written, and connect to `iquser`. Then issue the following command:

```
BACKUP DATABASE
TO '/dev/rmt/0n'
TO '/dev/rmt/1n'
WITH COMMENT 'Jan 18 full backup of iquser'
```

The catalog store is backed up first, to `/dev/rmt/0n`. The IQ store is backed up next, to both tapes.

Example 2 — Incremental Backup

To make an incremental backup of the same database, this time using only one tape device, issue the command as follows:

```
BACKUP DATABASE
INCREMENTAL
TO '/dev/rmt/0n' SIZE 150
WITH COMMENT 'Jan 30 incremental backup of iquser'
```

Other Examples

An example of how to restore this database from these two backups is provided later in this chapter.

For examples of backups that specify read-only files and dbspaces, see the BACKUP statement in *Reference: Statements and Options*.

Recovery from Errors During Backup

There are two likely reasons for a failed backup: insufficient space, or hardware failure.

Problems with third party software could also cause a failure.

Checking for Backup Space

BACKUP uses the **STACKER** and **SIZE** parameters to determine whether there is enough space for the backup.

- For disk backups, if it decides that you have not provided enough space, it fails the backup before actually writing any of the data.
- If it decides that there is enough space to start the backup, but then runs out before it finishes (for example, if your estimate is incorrect, or if a user in another application fills up a lot of disk space while your backup is in progress), an attended backup prompts you to load a new tape, or to free up disk space. An unattended backup fails if it runs out of space.
- If neither **STACKER** nor **SIZE** is specified, backup proceeds until it completes or until the tape or disk is full. If you run out of space, an attended backup prompts you to load a new tape, or to free up disk space; an unattended backup fails.

Recovery Attempts

If a backup fails, the backup program attempts to recover.

The recovery process is:

- If backup fails during either the checkpoint at the start of backup or the checkpoint when backup is complete, it performs normal checkpoint recovery.
- If backup fails between checkpoints, it rolls back the backup.

- If the system fails at any time between the initial and final checkpoint and you must restore the database, you must do so using an older set of backup tapes or disk files.
- If the system fails during the final checkpoint after a **FULL** backup, you can restore from the backup tapes or files you have just created.

After you Complete a Backup

In the event that you ever need to move a database or one of its dbspaces, you need to know the name of every dbspace in the database when the backup was made.

Sybase IQ includes a mechanism that verifies an existing Sybase IQ version 12.6 or later database backup using the **VERIFY** clause of the **RESTORE** SQL statement.

Performing Backups with Non-Sybase Products

Sybase IQ supports backup and restore using a number of third-party products. The package you use must conform to the Adaptive Server Enterprise Backup Interface. Check the documentation for your product to be sure that it supports Sybase databases.

To perform such a backup or restore, you issue the **BACKUP** or **RESTORE** statement as if you were using Sybase IQ to perform the operation, with the following exceptions:

- For each *archive_device*, instead of specifying the actual device name, specify a string in the following format:

  ```
  dll_name::vendor_specific_information
  ```
- Do not specify the **STACKER** or **SIZE** parameters.

The *dll_name* corresponds to a Dynamic Link Library loaded at run time. It can be from 1 to 30 bytes long, and can contain only alphanumeric and underscore characters. The *dll_name* must be the same for each *archive_device*.

The content of *vendor_specific_information* varies by product, and can differ for each *archive_device*. The total string (including *dll_name*:: and vendor information) can be up to 255 bytes long.

The backup program passes vendor information to the third-party program automatically. When you request a third-party backup, it places this information in the backup header file, and writes the header file on the first tape or disk file actually created for each *archive_device* you specify.

Note: Only certain third party products are certified with Sybase IQ using this syntax. See the *Release Bulletin* for additional usage instructions or restrictions. Before using any third party product to back up your IQ database in this way, make sure it is certified. See the Sybase Certification Reports for the Sybase IQ product in *Technical Documents*.

See also

- *Recording dbspace Names* on page 443
- *Restoring in the Correct Order* on page 434

- *Specifying Archive Devices* on page 414
- *The RESTORE Statement* on page 430
- *Verifying a Database Backup* on page 439

Virtual Backups

A virtual backup, sometimes called a NULL backup, backs up all of an IQ database except the IQ store table data.

You must make a separate operating system-level copy of the corresponding IQ store. To restore from a virtual backup, you must first restore the corresponding system-level copy of the IQ store and then proceed with the IQ full restore of the virtual backup.

A virtual backup backs up:

- All IQ catalog data
- All IQ metadata
- All metadata in the IQ store not specific to individual tables. (Includes the freelist, backup and checkpoint information.)

A virtual backup does not back up data or metadata from tables other than those mentioned above.

To make a virtual backup, specify either the **VIRTUAL DECOUPLED** or **VIRTUAL ENCAPSULATED** parameter in the **BACKUP** command when performing a full IQ Backup. The **VIRTUAL** parameters prevent IQ from copying table data and metadata in the IQ store to the backup file.

See also

- *Specifying Virtual Backup* on page 414

Types of Virtual Backups

There are two types of virtual backup.

- **Encapsulated virtual backup –** A restore of the system-level backup followed by a restore of the IQ virtual backup results in a fully restored database.
- **Decoupled virtual backup –** A restore of the system-level backup followed by a restore of the IQ virtual backup followed by an incremental-since-full restore results in a fully restored database.

Performing Encapsulated Virtual Backups

For the system-level backup of table data to be consistent with the virtual backup without additional steps, the system-level backup must be made during the backup command and by the backup transaction. T

he parameter `VIRTUAL ENCAPSULATED` *'shell-command'* allows arbitrary shell commands to be executed as part of the backup operation to guarantee these semantics. If the shell commands return a non-zero status, the backup operation returns an error. The user must guarantee that the shell commands correctly perform the system-level backup.

Use a SQL statement similar to the following:

```
BACKUP DATABASE FULL VIRTUAL ENCAPSULATED
'dd if=iqdemo.iq of=iqdemo.iq.copy'
TO 'iqdemo.full'
```

Restoring from Encapsulated Virtual Backup

Follow these steps to restore from encapsulated virtual backup.

1. Restore the system-level copy of the IQ store.
2. Perform a full IQ restore from the backup file.
3. Start the IQ database.

Performing Decoupled Virtual Backups

If the system-level backup is done outside the backup transaction, the IQ store backup will not be consistent with the IQ backup file.

However, a non-virtual IQ incremental backup together with the Virtual full backup will represent a consistent database. This is because the IQ incremental backup will copy all IQ store data and metadata that have changed during or since the Virtual full backup. Note that even the automatic commit and checkpoint that are part of the backup command modify the IQ store, making an independent system-level backup inconsistent. Trying to use the database without applying the incremental restore will give unpredictable results.

1. Perform a full IQ backup, using a SQL statement similar to the following:

   ```
   BACKUP DATABASE FULL VIRTUAL DECOUPLED
   TO 'iqdemo.full'
   ```

2. Perform a system-level backup of the IQ store with a shell command:

   ```
   dd if=iqdemo.iq of=iqdemo.iq.copy
   ```

3. Perform a non-virtual incremental IQ backup:

   ```
   BACKUP DATABASE INCREMENTAL SINCE FULL
   TO 'iqdemo.isf'
   ```

Restoring from a Decoupled Virtual Backup

Follow these steps to restore from a decoupled virtual backup.

1. Restore the system-level copy of the IQ store, for example:

```
dd if =iqdemo.copy of=iqdemo.iq
```

2. Restore from the IQ full backup file.

   ```
   RESTORE DATABASE iqdemo.db FROM 'iqdemo.full'
   ```

3. Restore from the IQ incremental backup file.

   ```
   RESTORE DATABASE iqdemo.db FROM 'iqdemo.isf'
   ```

4. Start the IQ database.

Virtual Backup with SAN Snapshot or Shadow Hardware

Storage Area Network (SAN) snapshot or shadow hardware provides more flexibility in the backup process by allowing the system-level backup to take place on the shadow copy rather than on the main database.

In place of the system-level copy that is part of the virtual backup, the shadow can instead be separated. A system-level backup can then be performed against the shadow copy of the IQ store. This allows the full backup to complete quickly.

System-Level Backups

The **BACKUP** command is the most reliable method you can use to back up IQ data. If you are careful to follow the correct procedures, you can use system-level backups for an IQ database.

You must follow these procedures when using system-level backups for backing up your IQ database. If you attempt to restore your IQ database files from a system-level backup without these safeguards in place, you are likely to cause data loss or inconsistency, either from activity in the database while the system-level backup occurred, or from missing files.

Shut Down the Database

Your IQ database must be shut down during a system-level backup. For details about performing system-level backup on multiplex systems, see Using Sybase IQ Multiplex.

You must shut down your IQ database before starting the system-level backup. You must also ensure that no one starts the IQ database until the system-level backup is complete.

Ensuring that the Database is Shut Down

The file protection of the `.db` file is read-only when the database is shut down cleanly, and set to read/write when the database is in use. If you are writing a script to perform backups, it is a good idea for the script to check the access mode of the file, to be sure that the database is shut down.

To ensure that a database remains shut down, the script can check the size of the `.iqmsg` file at the start and end of the script to make sure it has not changed. If the database was started while the script was running, the `.iqmsg` file is larger.

See also

- *Back Up the Right Files* on page 424
- *Restoring from a System-Level Backup* on page 425

Back Up the Right Files

Back up required files and optional files.

Required Files

You must back up the following files:

- All `SYSTEM` dbspace files, typically named `dbname.db`

  Note: There may be additional dbspaces in the catalog store, and are listed in `SYSDBSPACES`.

- The transaction log file, which is required for system recovery, typically named `dbname.log`
- The `IQ_SYSTEM_MAIN` dbspace file, typically named `dbname.iq`
- Files for any additional dbspaces that have been added to the IQ main store.

Save the lengths of the following files:

- The `IQ_SYSTEM_TEMP` dbspace file, typically named `dbname.iqtmp`
- Additional files that have been added to `IQ_SYSTEM_TEMP`

Backing up the temporary dbspaces is not required. IQ can reconstruct any temporary dbspace provided that it sees a file of the correct length at the time the database starts. Therefore, you may simply keep records of the sizes of the files or raw devices used to hold the temporary dbspaces.

Optional Files

Sybase recommends that you back up the ASCII message files such as `dbname.iqmsg` and the `$IQDIR15/logfiles/*.srvlog` and `$IQDIR15/logfiles/*.stderr` files, even though these files are not required for a restore. If problems occur during a restore, the `.iqmsg` file contains information that proves that the database was shut down before the backup started.

These files may be useful in diagnosing the cause of the database failure you are recovering from. Be sure to make a copy before restoring, for use in later analysis.

If IQ message log wrapping is enabled, you will probably want to back up the `.iqmsg` file so that all messages are accessible in the event you need them for diagnostic purposes.

If message log archiving is enabled (the **IQMsgMaxSize** server option or the **-iqmsgsz** server startup switch is not equal to zero and the **IQMsgNumFiles** server option or the **-iqmsgnum** server start up switch is not equal to zero), the server automatically backs up the message log archives. The maximum amount of message log that is archived is 128GB, which is sufficient in most cases.

Note: Backing up the message log archives is required before a server restart. After the server restarts, the existing log archives are ignored and a new archive is created when the `dbname.iqmsg` file is full. To preserve the old archive logs, back up the files before restarting the server.

Keeping Your Backup List Updated

It is critical to add to your system backup specification any dbspaces that are added to the database, whether they are in `SYSTEM`, `IQ_SYSTEM_MAIN`, or `IQ_SYSTEM_TEMP`. If a dbspace is added several months down the road, or after some turnover in your organization, you may miss this step.

To ensure that you are backing up all the files you need, use a script for system-level backups. In the script, before starting the backup, compare a select from *SYSFILE* (for the system dbspaces) and from *SYSIQFILE* (for the IQ dbspaces) to a list of dbspaces known to be in the system backup specification.

Raw Devices and Symbolic Links

If your database files are on raw devices, be sure your system backup is backing up the raw device contents, not just the name of the device in `/dev/*`.

If symbolic links are used for raw device names, as recommended, be sure the system backup utility follows the symbolic link and backs up the device.

See also

- *Restoring from a System-Level Backup* on page 425
- *Shut Down the Database* on page 423

Restoring from a System-Level Backup

If you must restore from a system-level backup, you must ensure that database server is shut down, just as it was during the backup.

When restoring a multiplex database, you must shut down all the secondary servers as well as the write server.

Ensuring that All Files Exist

Before restoring, review the table of contents of the backup to ensure that all files required for IQ are present. The list of files depends on your application.

In the case of the temporary dbspace files, ensure that files or raw devices are present with the correct file names (or symbolic links) and lengths. Contents of temporary dbspace files are irrelevant until the database restarts.

Checking Ownership and Permissions
Ensure that ownership and permission levels do not change during the system-level restore.

See also

- *Back Up the Right Files* on page 424
- *Shut Down the Database* on page 423

Validating Your Database

Backing up a database is useful only if the database is internally consistent.

Backup always makes sure that the database is in a usable state before proceeding. However, validating a database before you perform a backup is a good idea, to ensure that the database you restore is stable. The restore program does not check for inconsistencies in the restored data, since the database may not even exist.

To validate your database, issue the following command:

```
sp_iqcheckdb 'check database'
```

The **sp_iqcheckdb** stored procedure, in conjunction with server startup switches, is the interface to the IQ Database Consistency Checker (DBCC).

DBCC has different verification modes that perform increasing amounts of consistency checking. There are three modes for checking database consistency and one for resetting allocation maps. Each mode checks all database objects, if you specify 'database' as the target in the **sp_iqcheckdb** command string. Individual tables and indexes can also be specified in the command string. If you specify individual table names, all indexes within those tables are also checked.

The database option **DBCC_LOG_PROGRESS** instructs **sp_iqcheckdb** to send progress messages to the IQ message file. These messages allow you to follow the progress of the **sp_iqcheckdb** procedure as it executes.

You should run **sp_iqcheckdb** before or after backup, and whenever you suspect a problem with the database.

Validating a Multiplex Database
Run **sp_iqcheckdb** only on the write server of an IQ multiplex.

If you run **sp_iqcheckdb** on a multiplex secondary server, an error is returned.

Concurrency Issues for sp_iqcheckdb

When you run **sp_iqcheckdb** on an entire database, **sp_iqcheckdb** reads every database page in use.

This procedure consumes most of the database server's time, so that the I/O is as efficient as possible. Any other concurrent activities on the system run more slowly than usual. The CPU utilization of DBCC can be limited by specifying the **sp_iqcheckdb** parameter **resources resource-percent**, which controls the number of threads with respect to the number of CPUs.

If other users are active when you run **sp_iqcheckdb**, the results you see reflect only what your transaction sees.

See also

- *Data Protection* on page 405
- *Operating System Session Shutdown* on page 42
- *Resource Issues Running sp_iqcheckdb* on page 461
- *System Recovery* on page 385
- *System Recovery and Database Repair* on page 453
- *Ensure that your Database is Consistent* on page 409
- *Validating the Database After You Restore* on page 437

Restoring Your Databases

Once you have created a database and made a full backup, you can restore the database when necessary. Sybase IQ restores the database to its state as of the automatic **CHECKPOINT** at the start of the backup.

Before You Restore

Conditions must be met before you can restore a database.

- You must have DBA privileges.
- To restore read-only files or dbspaces from an archive backup, the database may be running and the administrator may connect to the database when issuing the **RESTORE** statement. The read-only file path name need not match the names in the backup, if they otherwise match the database system table information.

  The database must not be running to restore a FULL, INCREMENTAL SINCE FULL, or INCREMENTAL restore of either a READWRITE FILES ONLY or an all files backup.

  The database may or may not be running to restore a backup of read-only files. When restoring specific files in a read-only dbspace or read-only files in a read-write dbspace except for IQ_SYSTEM_MAIN, the dbspace must be offline. This restriction does not apply to IQ_SYSTEM_MAIN. Selective restore can be used to restore a read-only dbspace, as long as the dbspace is still in the same read-only state.

- To restore all files in a database or from a read-write files only backup, you must be connected to the `utility_db` database. For information on `utility_db` and how to set privileges for using it, see the *Installation and Configuration Guide* for your platform.
- When restoring all the files in the database or from a read-write files only backup, no user can be connected to the database being restored. **RESTORE** exits with an error if there are any active Read Only or Read/Write users of the specified database.
  Sybase recommends that you use two startup switches to restrict connections:
  - Use **-gd DBA** so that only users with DBA authority can start and stop databases on a running server. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
  - Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

  An alternate way to restrict connections is to specify

  ```
  sa_server_option('disable_connections', 'ON')
  ```

  just after you start the connection where you are restoring and

  ```
  sa_server_option('disable_connections', 'OFF')
  ```

  on the same connection after restoring. The disadvantage is that this method precludes emergency access from another DBA connection.
- You must restore the database to the appropriate server, and that server must have the archive devices you need. When you use the Sybase-provided restore, you need the same number of archive devices (that is, the disk files or tape drives) as when the backup was created.
- For a full restore, the store files (by default the `.iq` files), the catalog store (by default the `.db` file), and the transaction log (by default the `.log` file) must not exist in the location to which you are restoring. If any of these files exist, you must delete them or move them to a different directory before doing the full restore.
  When a full restore begins, it destroys all old database files and then recreates them. The requirement that you manually delete the store, catalog store, and transaction log files protects you from doing a full restore accidentally.
- For any incremental restore, the catalog store (`.db`) must exist. If it exists, but in a different location than the one you are restoring to, move database files. If it does not exist, you can only do a full restore. (If you do a full restore before any incremental restore, the correct files will be in place.)
- For any incremental restore, the database must not have changed since the last restore.

Restore requires exclusive access to the database and to the server. To gives the DBA greater control over inadvertent opens of the database, start the database server with the **-gd DBA** option set, but do not start the database you are restoring. **RESTORE** automatically starts the database in such a way that no other users can connect to it.

You must restore an entire backup or set of backups, including the full backup and all subsequent incremental backups. Restoring individual files from a backup archive is supported for read-only dbspaces and files alone. However, you can move database files to a new location using the **RENAME** clause of the **RESTORE** command.

Before you restore the database, you can verify the database backup (an existing Sybase IQ version 12.6 or later database backup) using the **VERIFY** clause of the **RESTORE** SQL statement.

See also

- *Command Line Options that Control Permissions* on page 30
- *Moving Database Files* on page 431
- *The Utility Database* on page 14
- *Utility Database Server Security* on page 314
- *Verifying a Database Backup* on page 439
- *SPACE ADMIN Authority Overview* on page 307

Restore Accommodates Dbspace Changes

During a set of incremental restores, **RESTORE** creates and drops dbspaces as needed to match what was done during the period of operation encompassed by the restores.

For example, assume that you make a full backup of a database, then add a dbspace to that database, and then do an incremental backup after adding the dbspace. When you restore from these backups, **RESTORE** creates a file for the new dbspace, at the start of the incremental restore. Similarly, if you drop a dbspace, it is dropped during the restore, although the actual file is not removed.

Note that the `file_name` column in the `SYSFILE` system table for the `SYSTEM` dbspace is not updated during a restore. For the `SYSTEM` dbspace, the `file_name` column always reflects the name when the database was created. The file name of the `SYSTEM` dbspace is the name of the database file.

Restoring Disk Backup Files

If you back up to disk and then move those files to tape, you must move them back to disk files with the same names as when you created the backup. Sybase IQ cannot restore disk files that are moved to tape directly from tape.

When you restore using the Sybase-provided backup and restore, you must specify the same number of archive devices (disk files) for the restore as were used to create the backup.

Restoring Tape Backup Files

When restoring from tape, you must position the tape to the start of the IQ data. **RESTORE** does not reposition the tape for you.

When you restore using the Sybase-provided backup and restore, you must use the same number of tape drives for the restore as were used to create the backup(s) you are restoring.

Specifying Files for an Incremental Restore

For an incremental restore, files you restore must match in number and size the files they replace, for both the IQ and catalog stores.

Keeping the Database Unchanged Between Restores

If you are doing a set of incremental restores, and any user changes the database before you finish restoring the complete set, the Restore program does not let you restore the remaining incrementals.

For example, if you have a set consisting of a full restore and two incrementals, and a user's write transaction commits after the full restore but before you issue the second or third **RESTORE** command, you cannot proceed with the incremental restores. Instead, you must restore the full backup and apply the incrementals again.

If the database has changed since the last restore and you try to do an incremental restore, the following error occurs:

```
Database has changed since the last restore
```

Note: Sybase IQ does not let you do an incremental restore if the database has changed since the previous restore. However, it does not prevent users from making changes. It is the responsibility of the DBA or system operator to ensure that no changes are made to the database until all restores are complete.

Restoring from a Compatible Backup

RESTORE lets you restore database files for Sybase IQ 15.0 and later. Due to changes in the format of the database, you cannot restore from a backup created on a 12.x version.

To move your data from a Sybase IQ 12.x database to Sybase IQ:

1. Upgrade to 12.6 ESD #11or later using the migration procedure described in the version 12.6 *Installation and Configuration Guide*.
2. Follow the migration procedure described in the 15.3 *Installation and Configuration Guide*.

RESTORE does not let you restore an Sybase IQ backup to an SQL Anywhere database.

The RESTORE Statement

To restore a database, use the **RESTORE** statement.

You must be connected to the `utility_db` database as DBA to issue this statement.

You must specify the *db_file* and at least one *archive_device*.

For *db_file* you specify the location of the catalog store file for the database (created with the suffix `.db` by default). You can specify the full path name or a path name relative to the directory where the database was created. If you specify a new path name, the catalog store and any files created relative to it are moved to that location, except for any files you include in a **RENAME** clause.

Just as for backup, each *archive_device* specifies the API (third party) and, for the Sybase API, the physical tape device or disk file name from which you are restoring. For third-party APIs, the content of the *archive_device* string depends on your vendor. The archive device must not be a raw disk device. When you restore from disk files using the Sybase API, you must supply the same number of archive devices as were specified when this backup was created.

Warning! If you misspell a tape device name and give a name that is not a valid tape device on your system, **RESTORE** assumes it is a disk file and tries to read from it.

Note: If you are restoring from tape devices on Windows, note that you do not need to redouble the backslashes when you specify tape devices for restore, as you did for backup.

Example 1 — Restoring to the Same Location

This Windows example restores a database to `iquser.db`. The database is restored from two disk files. All database files are restored to their original locations.

```
RESTORE DATABASE 'iquser.db'
FROM 'c:\\iq\\backup1'
FROM 'c:\\iq\\backup2'
```

See also

- *Performing Backups with Non-Sybase Products* on page 420
- *Recording dbspace Names* on page 443
- *Restoring in the Correct Order* on page 434
- *Specifying Archive Devices* on page 414
- *Verifying a Database Backup* on page 439

Moving Database Files

Move database files to a new location.

If you need to move database files to a new location—for example, if one of your disk drives fails—you use one of the following methods:

- To move the database file that holds the catalog store (by default, the `.db` file), you simply specify the new name as *db_file*.
- To move or rename the transaction log file, use the `dblog` transaction log utility. For syntax and details, see the *Utility Guide > dblog Database Administration Utility*.
- To move any other database file, you use the **RENAME** option.

Restoring to a Raw Device

When restoring to a raw device, make sure that the device is large enough to hold the dbspace being restored. IQ **RESTORE** checks the raw device size and returns an error, if the raw device is not large enough to restore the dbspace.

The operating system takes a small amount of space on the raw device and the IQ dbspace occupies the rest. When you restore the dbspace, your raw partition must hold both the IQ dbspace and the space reserved for the operating system.

To restore an IQ main or temporary dbspace to a raw partition, find the raw device size needed for each IQ dbspace from system tables as follows:

```
SELECT segment_type, file_name, block_count,
data_offset, block_size,
(block_count * block_size) + data_offset AS raw_size
FROM SYS.SYSIQFILE, SYS.SYSIQINFO
where segment_type !='Msg' ORDER BY 1,2
```

The `segment_type` and `file_name` are informational. Segments of type ‘Main’ or ‘Temp’ may be stored on a raw partition, but message files (type ‘Msg’) may not. The `file_name` is the name of the dbspace.

The `block_count` column is an integer, the number of blocks used by IQ.

The `data_offset` column is an integer, the number of bytes reserved for the operating system.

The `block_size` column is an integer, the number of bytes per IQ block.

The `raw_size` column is an integer, the minimum size in bytes of a raw device needed to restore this dbspace. Sybase recommends restoring to a raw device that is at least 10MB larger than the original raw device.

Example 2 — Moving the Catalog Store

This example restores the same database as Example 1.

In Example 2, however, you move the catalog store file and any database files that were created relative to it. To do so, you replace the original file name with its new location, `c:\newdir`, as follows:

```
RESTORE DATABASE 'c:\\newdir\\iqnew.db'
FROM 'c:\\iq\\backup1'
FROM 'c:\\iq\\backup2'
```

Sybase IQ moves database files other than the catalog store as follows:

- If you specify a **RENAME** clause, the file is moved to that location.
- If you do not specify a **RENAME** clause, and the file was created using a relative path name, it is restored relative to the new location of the database file. In other words, files originally created relative to the `SYSTEM` dbspace, which holds the catalog store file, are restored relative to the catalog store file. Files originally created relative to the catalog store are restored relative to the catalog store.
- If you do not specify a **RENAME** clause, and the file was created using an absolute path name, the file is restored to its original location.

In other words, if you want to move an entire database, you should specify in a **RENAME** clause the new location for *every* IQ dbspace in the database—required, temporary, and user-defined. The SYSTEM dbspace is the only one you do not include in a **RENAME** clause.

If you only want to move some of the files, and overwriting the original files is not a problem, then you only need to rename the files you actually want to move.

You specify each dbfile name as it appears in the `SYSIQDBFILE` table. You specify *new_dbspace_path* as the new raw partition, or the new full or relative path name, for that dbspace.

You cannot use the **RENAME** option to specify a partial restore.

Relative path names in the **RENAME** clause work as they do when you create a database or dbspace: the main IQ store dbspace, temporary store dbspaces, and message log are restored relative to the location of `db_file` (the catalog store); user-created IQ store dbspaces are restored relative to the directory that holds the catalog store.

If you are renaming files while restoring both full and incremental backups, be sure you use the dbspace names and paths consistently throughout the set of restores. It is the safest way to ensure that files are renamed correctly.

If a dbspace was added between the full backup and an incremental backup, and you are renaming database files, you need one more **RENAME** clause for the incremental restore than for the full restore. Similarly, if a dbspace was deleted between backups, you need one fewer **RENAME** clause for the restores from any backups that occurred after the dbspace was deleted.

Example 3 — Moving a User dbspace

This example shows how you restore the full and incremental backups in the example shown earlier in this chapter. In this case, media failure has made a UNIX raw partition unusable. The user-defined dbfile on that raw partition, `IQ_USER`, must be moved to a new raw partition, /dev/rdsk/c1t5d2s1. No other database files are affected.

First, you connect to the `utility_db` database. Then you restore the full backup from two tape devices. In this case they are the same two tape devices used to make the backup, but the devices could differ as long as you use the same number of archive devices, the same media type (tape or disk), and the same tape sets in the correct order.

The first **RESTORE** command is:

```
RESTORE DATABASE 'iquser'
FROM '/dev/rmt/0n'
FROM '/dev/rmt/1n'
RENAME IQ_SYSTEM_MAIN TO '/dev/rdsk/c2t0d1s1'
RENAME IQ_SYSTEM_TEMP TO '/dev/rdsk/c2t1d1s1'
RENAME IQ_SYSTEM_MSG TO 'iquser.iqmsg'
RENAME IQ_USER TO '/dev/rdsk/c1t5d2s1'
```

The second **RESTORE** command, to restore the incremental backup, is:

```
RESTORE DATABASE 'iquser'
FROM '/dev/rmt/0n'
RENAME IQ_SYSTEM_MAIN TO '/dev/rdsk/c2t0d1s1'
RENAME IQ_SYSTEM_TEMP TO '/dev/rdsk/c2t1d1s1'
RENAME IQ_SYSTEM_MSG TO 'iquser.iqmsg'
RENAME IQ_USER TO '/dev/rdsk/c1t5d2s1'
```

Note: You could also issue these commands with only the last **RENAME** clause, since only one dbspace is being restored to a new location. Listing all of the files or raw partitions, as shown here, ensures that you know exactly where each will be restored.

See also

- *Before You Restore* on page 427
- *Command Line Options that Control Permissions* on page 30
- *The Utility Database* on page 14
- *Utility Database Server Security* on page 314
- *Verifying a Database Backup* on page 439

Displaying Header Information with CATALOG ONLY Option

The **CATALOG ONLY** option displays the header information for the database, placing it in the `.backup.syb` file.

It does not restore any data, either from the catalog store or the IQ store.

When you specify **CATALOG ONLY** you must include the **FROM archive_device** clause, but omit the **RENAME** clause.

See also

- *Content of the Backup Log* on page 442
- *Displaying Header Information* on page 438
- *Getting Information about Backups and Restores* on page 441

Adjusting Data Sources and Configuration Files

When you move a database, you may need to modify your data sources, configuration files, and integrated logins to reflect the new location of the database.

Restoring in the Correct Order

When you restore from a full backup, every block in use at the time the backup was made is written to disk. When you restore from an incremental backup, only the blocks that changed between the previous backup (or the previous full backup) and this backup are written to disk.

You must restore full and incremental backups in the correct order, with a separate **RESTORE** command for each backup you are restoring. **RESTORE** ensures that backups are restored in order, and gives the following error if it determines that the order is incorrect:

```
SQL Code:  -1012009
```

```
SQL State: QUA09
```

```
This restore cannot immediately follow the previous restore.
```

To determine the correct order, you need the information about backup files that is stored in the backup log.

Restore backups as follows:

- If your database is inconsistent, or if you are moving any files to a new location, you must restore a **FULL** backup.
- If your most recent backup is a **FULL** backup, or if you need to restore a database to the state that existed before any existing incremental(s) were made, restore the full backup only.
- If you have an **INCREMENTAL_SINCE_FULL** backup that precedes the database failure, first restore from the last **FULL** backup, and then restore the **INCREMENTAL_SINCE_FULL** backup.
- If you do not have an **INCREMENTAL_SINCE_FULL** backup, but you have performed one or more **INCREMENTAL** backups since your last **FULL** backup, first restore the **FULL** backup, and then restore the **INCREMENTAL** backups in the order in which they were made.

You can also use the advisory stored procedure **sp_iqrestoreaction** to suggest the sequence of restore actions required to attain a stable database set. Always confirm the steps suggested against above rules. The stored procedure also does not factor in moving any database files.

Within a given backup, the order in which you restore tapes is also important. In particular, you need to keep track of the order of tapes in each backup tape set, that is, the set of tapes produced in a given backup on a given archive device:

- You must restore the tape set that contains the backup of the catalog store first, and it must be on the first archive device.
- Within each set, you must restore tapes in the order in which they were created.
- You cannot interleave sets; each set must be restored before you can restore another set.
- After the first set, the order in which sets are restored does not matter, as long as it is correct within each set.

Use the same number of drives to restore as were used to produce the backup, so that you do not accidentally interleave tapes from different sets.

Example

Assume that you are restoring a full backup, in which you used three archive devices, and thus produced three tape sets, A, B, and C.

The contents of each set, and the restore order, are as follows:

- **Set A –** Tapes A_1, A_2, and A_3. Tapes A_1 and A_2 contain the catalog store. This set must be restored first, and must be in the first device.

- **Set B –** Tapes B_1 and B_2. These must be restored as a set, after Set A, and either before or after Set C. They can be in either the second or third device.
- **Set C –** Tapes C_1, C_2, and C_3. These must be restored as a set, after Set A, and either before or after Set B. They can be in either the second or third device.

The Restore program checks that tapes within each set are in the correct order on a single device. If not, you get an error, and the restore does not proceed until you supply the correct tape. Except for the set with the catalog store, it does not matter which set you put on a given device.

Note: You must ensure that the catalog store tape set is restored first. The Restore program does not check this.

Although these rules also apply to disk files, you are not likely to back up to multiple files on a given disk device.

See also

- *Distribution of Backup Data* on page 408
- *Getting Information about Backups and Restores* on page 441
- *Performing Backups with Non-Sybase Products* on page 420
- *Recording dbspace Names* on page 443
- *Specifying Archive Devices* on page 414
- *The RESTORE Statement* on page 430
- *Verifying a Database Backup* on page 439

Reconnecting After You Restore

Sybase IQ requires the DBF parameter and database file name in order to connect to a database under certain circumstances.

This situation occurs when you use Interactive SQL and you have restored that database from backup while connected to `utility_db`.

For example, include the DBF parameter as follows:

```
CONNECT USING 'uid=DBA;pwd=sql;dbf=node1/users/fiona/mydb.db;
links=tcpip{host=serv1;port=1234};eng=serv1_iqdemo'
```

Prior to Sybase IQ 12.6 ESD5, you could connect to a restored database using the following syntax:

```
CONNECT DATABASE mydb USER DBA IDENTIFIED BY SQL
```

The preceding command now returns a `specified database not found` error.

Another way to avoid the error is to enter a **START DATABASE** command while connected to `utility_db`, for example:

```
START DATABASE mydb
```

Use this method when connecting via Interactive SQL.

See also

- *Database Connection Issues* on page 492

Renaming the Transaction Log after you Restore

When you rename or move all other files in the database, you should also do the same for the log file.

To move or rename the log file, use the Transaction Log utility (**dblog**). You should run this utility:

- After using **RESTORE** with a new database name
- After using **RESTORE** with the **RENAME** option

Note: The database server must not be running on that database when the transaction log file name is changed.

You can also use **dblog** to rename the transaction log, even if you have not restored the database, given certain restrictions. You can access the Transaction Log utility from the system command line, using the **dblog** command-line utility. See *Utility Guide > dblog Database Administration Utility*.

Validating the Database After You Restore

To ensure that tapes have been restored in the correct order, you should run the stored procedure **sp_iqcheckdb** after you finish restoring your database.

If you are restoring a set of incremental backups, it is safest to run **sp_iqcheckdb** after restoring each backup. To save time, however, you may prefer to run **sp_iqcheckdb** only after restoring the last incremental.

See also

- *Ensure that your Database is Consistent* on page 409
- *Validating Your Database* on page 426

Restore Requires Exclusive Write Access

Once **RESTORE** starts, no other users are allowed to access the specified database.

If you restore from a full backup and then from one or more incremental backups, you should ensure that no users are modifying the database between the restores. The modifications are permitted, but you cannot perform any more incremental restores. Instead, you must start the entire restore again.

This restriction extends to any incremental restores you may need if your system crashes during recovery. If you need to recover from a system or media failure that occurs during a restore, you must do one of the following:

- Continue the original sequence of full and incremental restore operations, or

- Perform a full restore, followed by any incremental restores needed to fully recover your database.

The default database server startup setting **-gd DBA** makes DBA privileges a requirement for starting up a database. When the DBA runs **RESTORE**, the command automatically starts the database, gets the information it needs for the restore, and then stops the database. At the end of the restore, the command starts the database, issues a checkpoint, and stops it again. This procedure ensures that the DBA has exclusive write access throughout a restore.

When all incremental restores are complete, the DBA issues the **START DATABASE** command again to allow other users access to the database.

To restore a multiplex database, see *Using Sybase IQ Multiplex*.

Displaying Header Information

You can display the contents of the header file by using the **RESTORE** statement with the **CATALOG ONLY** option and no **FILE** clauses.

A **RESTORE** with **CATALOG ONLY** produces the information in the same format as the backup log entry for an actual **RESTORE**.

You can get more information about the backup archive using the command-line utility **db_backupheader**, which accepts the file path corresponding to the first backup archive. The utility reads the backup archive file. It does not connect to the database.

The backup archive information includes:

- backup information
- database information at the time of the backup
- dbspace information for each dbspace in the database
- dbfile information for each dbfile in the dbspaces

See also

- *Content of the Backup Log* on page 442
- *Displaying Header Information with CATALOG ONLY Option* on page 434
- *Getting Information about Backups and Restores* on page 441

Recovery from Errors During Restore

If an incremental restore fails early in the operation, the database is still usable (assuming it existed and was consistent before the restore).

If a full restore fails, you do not have a usable database.

If a failure occurs after a certain point in the operation, the restore program marks the database as inconsistent. In this case recovery is only possible by means of a **FULL RESTORE**. If you were performing a **FULL RESTORE** when the failure occurred, you may need to go back to the previous **FULL BACKUP**.

Verifying a Database Backup

Sybase IQ includes a mechanism that verifies an existing Sybase IQ version 12.6 or later database backup using the **VERIFY** clause of the **RESTORE** SQL statement.

The verification process directs the server to validate the specified Sybase IQ database backup archives for a full, incremental, incremental since full, or virtual backup, and to check the specified archive for the same errors a restore process checks, but performs no write operations. All status and error messages are written to the server log file.

The backup verification process can run on a different host than the database host. You must have DBA, BACKUP, or OPERATOR authority to run **RESTORE VERIFY**.

For syntax and usage of the **RESTORE VERIFY** clause, see the **RESTORE** statement in *Reference: Statements and Options*.

Note: The verification of a backup archive is different than the database consistency checker (DBCC) verify mode (`sp_iqcheckdb 'verify...'`). **RESTORE VERIFY** validates the consistency of the backup archive to be sure it can be restored, whereas DBCC validates the consistency of the database data.

Run `sp_iqcheckdb 'verify...'` before taking a backup. If an inconsistent database is backed up, then restored from the same backup archive, the data continues to be in an inconsistent state, even if **RESTORE VERIFY** reports a successful validation.

Verification of an Incremental Backup

Use the **RESTORE… VERIFY COMPATIBLE** clause to check the compatibility of an incremental archive with the existing database files. If the database files do not exist on the system on which you invoke **RESTORE…VERIFY COMPATIBLE**, an error is returned. If you specify **COMPATIBLE** while verifying a full backup, the keyword is ignored; no compatibility checks need to be made while restoring a full backup.

If you specify **RESTORE VERIFY** without **COMPATIBLE** for an incremental restore, Sybase IQ does not look for any dbspaces and does not perform any compatibility checks. No warning is reported, even if the files do not exist. The compatibility check is performed only when you include the **COMPATIBLE** clause.

If you specify **RESTORE VERIFY COMPATIBLE** for an incremental restore, and the IQ catalog store or any of the Sybase IQ dbspaces do not exist, the compatibility check cannot be made; an error is reported and the operation fails.

During validation of an incremental backup, the **RESTORE VERIFY COMPATIBLE** process opens the Sybase IQ dbspaces in read-only mode to perform consistency checking. No dbspaces are modified.

In incremental restores, if the database has been modified or the particular incremental archive is not the correct archive for the database, **RESTORE VERIFY COMPATIBLE** reports the error `Database has changed since last restore.` (SQLCODE -1012008, SQLSTATE QUA08) or `This restore cannot immediately follow the previous restore.` (SQLCODE -1012009, SQLSTATE QUA09).

Verification Progress Reporting

The **RESTORE VERIFY** process verifies every stripe specified in the command. As the verification process checks stripes and their corresponding files, it reports progress in terms of the number of IQ blocks verified. After every 5000 IQ blocks verified, a message is written to the server log file:

```
5000/100000 (5%) Blocks verified
```

A final message is written to the server log file when 100% of the IQ blocks are verified.

The messages `RESTORE VERIFY Started`, the number of IQ blocks to be verified, and `RESTORE VERIFY successfully completed` are also written in the server log, when the verify action starts and completes, respectively. For example:

```
I. 11/17 06:45:24. VERIFY RESTORE Started
I. 11/17 06:45:24. Total number of IQ blocks to be verified: 764
I. 11/17 06:45:24. Total number of IQ blocks verified: 764/764 ( 100
% )
I. 11/17 06:45:24. VERIFY RESTORE Successfully Complete
```

Verification Error Reporting

If the verification process finds errors after which it can continue, the process continues checking the archive and logs information for the errors detected.

The errors for which the verification can continue are:

- `Header of block to be restored appears to be corrupted. (SQLCODE -1012011l, SQLSTATE QUA11)`
- `Media data appears corrupted (bad checksum). (SQLCODE -1012012, SQLSTATE QUA12)`
- `Media meta data appears corrupted (boundary record). (SQLCODE -1012013, SQLSTATE QUA13)`
- `Media meta data appear corrupted (multiple begin boundary records). (SQLCODE -1012014, SQLSTATE QUA14`

If any of these errors are found and the verification process can continue to the end of the archive, Sybase IQ reports this error:

`The verification of the provided archive has failed. Please check the server log for details of the errors thrown during verify.`

If any error pertaining to **RESTORE** is found other than the errors above, the error that occurred is reported, and the verification process stops.

See also

- *Before You Restore* on page 427
- *Command Line Options that Control Permissions* on page 30

- *Moving Database Files* on page 431
- *The Utility Database* on page 14
- *Utility Database Server Security* on page 314
- *Performing Backups with Non-Sybase Products* on page 420
- *Recording dbspace Names* on page 443
- *Restoring in the Correct Order* on page 434
- *Specifying Archive Devices* on page 414
- *The RESTORE Statement* on page 430

Backups and Symbolic Links (UNIX Only)

In backups involving symbolic links, Sybase IQ may create dbspaces in a directory other than the one desired.

For example, suppose that you create dbspaces in the following files:

```
-rw-r--r--   1 fiona  sybase   122880000 Feb 26 18:27 iqdemo.db
-rw-r--r--   1 fiona  sybase   122880000 Feb 26 18:27 iqdemo.iq1
-rw-r--r--   1 fiona  sybase   122880000 Feb 26 18:27 iqdemo.iq2
-rw-r--r--   1 fiona  sybase   122880000 Feb 26 18:27 iqdemo.iq3
-rw-r--r--   1 fiona  sybase   122880000 Feb 26 18:27 iqdemo.iqtmp
-rw-r--r--   1 fiona  sybase   122880000 Feb 26 18:27 iqdemo.iqmsg
```

If you create the following links first, the dbspaces will be created in the directories (or on the raw partitions) to which the links point:

```
lrwxrwxrwx   1 fiona  sybase        14 Feb 26 17:48 iqdemo.iq1 ->
 LINKS/iqdemo.iq1
```

```
lrwxrwxrwx   1 fiona  sybase        14 Feb 26 17:48 iqdemo.iq2 ->
 LINKS/iqdemo.iq2
```

```
lrwxrwxrwx   1 fiona  sybase        18 Feb 26 17:48 iqdemo.iq3 ->
 /dev/rdsk/c2t6d0s0
```

When you back up the files and restore them with the **CATALOG ONLY** option, you don't see anything telling you that these files were links; in fact, this information is not saved.

Sybase IQ saves these files as though they were actually present in the directory where the symbolic links reside. When you do the restore, the files are recreated in the directories or on the raw partitions named by the database name. Whether or not the links exist at restore time, they are never used again. The database is restored to its original location.

Getting Information about Backups and Restores

Sybase IQ provides a backup log, `.backup.syb`, to help you manage your backup media.

This log is not used to create the backup or to restore the database; however, information describing the backup or restore is recorded in this file during both Backup and Restore.

Note: To display only the information about a particular backup, you can run **RESTORE** with the **CATALOG ONLY** option. This option displays the header file for a backup from the media rather than from the file, so that the DBA can identify what is on the tape or file.

See also

- *Distribution of Backup Data* on page 408
- *Restoring in the Correct Order* on page 434
- *Content of the Backup Log* on page 442
- *Displaying Header Information with CATALOG ONLY Option* on page 434
- *Displaying Header Information* on page 438

Locating the Backup Log

The `.backup.syb` file is in ASCII text format.

Its location depends on the setting of environment variables at the time the server is started:

- On UNIX, the server tries to place it in the following locations, in this order:
  - The directory specified by the IQLOGDIR15 environment variable.
  - The directory specified by the HOME environment variable.
  - The home directory as obtained from account information.
  - The current directory (where the server was started).

  If the file is placed in the home directory, it is prefixed with a "." in order to make it a hidden file. If the file is placed in the current directory, it is not prefixed.
- On Windows, the server tries to place it in the following locations, in this order:
  - The directory specified by the IQLOGDIR15 environment variable.
  - The directory that holds the server executable files.

Content of the Backup Log

For every backup or restore you perform, the backup log contains a comma-separated list of fields.

The backup log provides:

- Operation (Backup or Restore)
- Version
- Database name
- Database type (Sybase IQ or SQL Anywhere)
- Date and time of backup or restore
- Creator user ID
- Type of backup/restore: Full, Incremental, or Incremental_since_full, or Database File Only (for SQL Anywhere databases only)
- Method: Archive (for IQ or Anywhere databases) or Image (Anywhere databases only)

- Location
- Comment (if entered on the **BACKUP** command), enclosed in single quotes. If the comment includes quotes, they appear as two consecutive single quotes.

Here is a sample backup log.

```
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 16:25:00.000', DBA,
Full, Arch, TED_FULL00, ''
```

```
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 16:53:00.000', DBA,
Incr, Arch, TED_X_bkup_inc, ''
```

```
RESTORE, 2.0, all_types.db, ASIQ, '2009-01-31 16:25:00.000', DBA,
Full, Arch, TED_FULL00, ''
```

```
RESTORE, 2.0, all_types.db, ASIQ, '2009-01-31 16:53:00.000', DBA,
Incr, Arch, TED_X_bkup_inc, ''
```

```
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 20:07:00.000', DBA,
InSF, Arch, A_partial2_yes_sf, ''
```

```
BACKUP, 2.0, all_types.db, ASIQ, '2009-01-31 20:07:00.000', DBA,
InSF, Arch, A_partial2_yes_sf, ''
```

See also

- *Displaying Header Information with CATALOG ONLY Option* on page 434
- *Displaying Header Information* on page 438
- *Getting Information about Backups and Restores* on page 441

Maintaining the Backup Log

It's a good idea to clean up the backup log after you purge backup media. Use a text editor to do so.

Be careful with your edits: once **BACKUP** or **RESTORE** records information in this file, it does not check its accuracy.

There is only one backup log on a server. The server must be able to read and write this file. The system administrator may want to limit access to this file by other users. If you are running more than one database server on a system, you should set the IQLOGDIR15 environment variable differently for each server, to produce separate backup logs.

Warning! Do not edit the backup log while a backup or restore is taking place. If you are modifying the file while **BACKUP** or **RESTORE** is writing to it, you may invalidate the information in the file.

Recording dbspace Names

In the event that you ever need to use the **RENAME** option of **RESTORE** to move a database or one of its dbspaces, you need to know the name of every dbspace in the database.

The dbspace names are in the `SYSFILE` table of every database, but you do not have this table available when you are restoring. Run **db_backupheader** on the first backup archive file path to view this information. Alternatively, you may execute the **sp_iqdbspace** and **sp_iqfile** stored procedures or issue the following statement any time you back up your database:

```
SELECT dbf.dbfile_name, f.*
FROM SYSFILE f, SYSDBFILE dbf
WHERE f.file_id=dbf.dbfile_id
```

Keep the results of this query some place other than the disk where the database resides, so that you have a complete list of dbspace names if you need them.

You can also run the following script in Interactive SQL. This script produces an output file that contains the set of rename clauses you use, if you do not actually change the location of any files. You can substitute any new file locations, and use the resulting file in your **RESTORE** statement.

Note: Because the database may not exist when you need to restore, you may want to run this script after you back up your database.

```
-- Get dbspace and IQ file names and add
-- rename syntax including quotation marks

select 'rename' as 'restore ... rename' ,
dbf.dbfile_name as 'IQ file' , 'to' as 'to' ,
'''' + f.file_name + '''' as 'file_path'
from SYSFILE f, SYSDBFILE dbf
where f.store_type=2 and f.file_id=dbf.dbfile_id

 -- Send output to a file in proper format
 -- without delimiters or extra quotation marks

output to restore.tst delimited by ' ' quote '';

 -- This produces a restore.tst file like the following:
 -- rename IQ_SYSTEM_MAIN to '/dev/rdsk/c2t0d1s7'
 -- rename IQ_SYSTEM_TEMP to '/dev/rdsk/c2t1d1s7'
 -- rename IQ_SYSTEM_MSG to 'all_types.iqmsg'
```

See also

- *Performing Backups with Non-Sybase Products* on page 420
- *Restoring in the Correct Order* on page 434
- *Specifying Archive Devices* on page 414
- *The RESTORE Statement* on page 430
- *Verifying a Database Backup* on page 439

Determining Your Data Backup and Recovery Strategy

To develop an effective strategy for backing up your system, you need to determine the best combination of full, incremental, and incremental-since-full backups for your site, and then set up a schedule for performing backups.

Consider the performance implications of various backup options, and how they affect your ability to restore quickly in the event of a database failure.

Scheduling Routine Backups

Make a full backup of each database just after you create it, to provide a base point, and perform full and incremental backups on a fixed schedule thereafter. It is especially important to back up your database after any large number of changes.

Your backup plan depends on:

- The load on your system
- The size of your database
- The number of changes made to the data
- The relative importance of faster backups and faster recovery

See also

- *Specify the Type of Backup* on page 413

Determining the Type of Backup

When you decide whether to do a full, incremental, or incremental_since_full backup, you need to balance the time it takes to create the backup with the time it would take to restore.

You also should consider media requirements. A given incremental backup is relatively quick and takes a relatively small amount of space on tape or disk. Full backups are relatively slow and require a lot of space.

Incremental_since_full is somewhere in between. It starts out as equivalent to incremental, but as the database changes and the number of backups since a full backup increases, incremental_since_full can become as time-consuming and media-consuming as a full backup, or worse.

In general, the opposite is true for restore operations. For example, if you need to restore from a very old full backup and a dozen or more incrementals, the restore may take longer and the backup may use up more space than a new full backup.

The obvious advantage of incremental backups is that it is much faster and takes less space to back up only the data that has changed since the last backup, or even since the last full backup, than to back up your entire database. The disadvantage of relying too heavily on incremental backups is that any eventual restore takes longer.

For example, once you have a full backup of your database, in theory you could perform only incremental backups thereafter. You would not want to do this, however, because any future recovery would be intolerably slow, and would require more tape or disk space than doing a full backup periodically. Remember that other users can have read and write access while you do backups, but no one else can use the database while you are restoring it. You might find yourself needing to restore dozens of incremental backups, with your system unavailable to users throughout the process.

A much better approach is a mix of incremental and full backups.

The greater the volume of your database changes, the more important it is do a backup, and the smaller the advantage of incremental backups. For example, if you update your database nightly with changes that affect 10 percent or more of the data, you may want to do an incremental_since_full backup each night, and a full backup once a week. On the other hand, if your changes tend to be few, a full backup once a month with incrementals in between might be fine.

Designating Backup and Restore Responsibilities

Many organizations have an operator whose job is to perform all backup and recovery operations. Anyone who is responsible for backing up or restoring an Sybase IQ database must have DBA privileges for the database.

Improving Performance for Backup and Restore

The overall time it takes to complete a backup or restore a database depends largely on the strategy you choose for mixing full and incremental restores.

Several other factors also affect the speed of backup and restore operations: the number of archive devices, data verification, the memory available for the backup, and size of the IQ and catalog stores.

Increasing the Number of Archive Devices

The **TO** clause in the **BACKUP** statement controls the number of archive devices.

Eliminating Data Verification

You can also improve the speed of backup and restore operations by setting **CRC OFF** in the **BACKUP** command.

This setting deactivates cyclical redundancy checking. With **CRC ON**, numbers computed on backup are verified during any subsequent restore operation, affecting performance of both commands. The default is **CRC ON**. If you turn off this checking, remember that you are giving up a greater assurance of accurate data in exchange for faster performance.

Spooling Backup Data

You may find that it is faster and more efficient to create backups on disk, and then spool them onto tape for archival storage. If you choose this approach, you need to unspool the data onto disk before restoring it.

Increasing Memory Used During Backup

The amount of memory used for buffers during backup directly affects backup speed, primarily for tape backups. The **BLOCK FACTOR** parameter of the **BACKUP** command controls the amount of memory used. If your backups are slow, you may want to increase the value of **BLOCK FACTOR** for faster backups.

The effect of **BLOCK FACTOR** depends on your operating system, and on the block size specified when the database was created. The default IQ page size of 128KB for newly created databases results in a default block size of 8192 bytes.

On UNIX, the default **BLOCK FACTOR** is 25. Sybase recommends setting **BLOCK FACTOR** to at least 25. With this combination, **BACKUP** is able to buffer data ideally for most UNIX tape drives, with enough data in memory that drives are kept busy constantly throughout the backup.

On Windows, the default **BLOCK FACTOR** is computed based on the database block size. This value usually achieves maximum throughput on Windows. Because of the way Windows handles tape devices, you may not be able to achieve faster backups by increasing the **BLOCK FACTOR**.

Balancing System Load

Sybase IQ allows you to perform backups concurrently with all other read/write operations, except those that affect the structure of the database.

It is still a good idea to schedule backups during times of low system use, however, to make the best possible use of system resources—disk, memory, and CPU cycles.

Controlling the Size of the Catalog Store

An IQ database consists of an IQ store and an underlying catalog store.

BACKUP makes a full backup of the catalog store at the start of every backup, both full and incremental. Ordinarily the catalog store is quite small, containing only the system tables, metadata, and other information Sybase IQ needs to manage your database. However, it is possible to create non-IQ tables in the catalog store. You can improve IQ backup performance by keeping any non-IQ data in a separate SQL Anywhere-only database, rather than in the catalog store.

Backup copies only the latest committed version of the database. Other version pages used by open transactions are not backed up.

Archiving Data with Read-Only Hardware

Recent regulations such as the Health Insurance Portability and Accountability Act (HIPAA) and Sarbanes-Oxley Act specify rigid rules for data retention and compliance, requiring data archived in a immutable, easily accessible form.

Data volumes may extend into the several terabyte range, while data retention periods range from a few years to decades.

WORM disk storage solutions evolved to address these requirements. WORM (write once, read many) storage began as optical disk technology allowing only one permanent write of each storage location. WORM disk arrays are known as *read-only hardware* in Sybase IQ. Read-only hardware functionality is provided by low-cost disk array hardware with a WORM protection layer added. The protection layer allows normal read-write use of the disks until the data is “frozen”.

When data is frozen, the user specifies an indefinite or fixed retention period. The disks may be frozen at the volume or file level. Once frozen, data may not be modified, and the retention period may be extended, but never decreased.

Read-only hardware functionality is not limited to WORM disk array hardware; you may also remove write privilege from a raw device or file system file after the dbspace is altered read-only.

See also

• *Select Archive Devices* on page 409

Using Read-Only Hardware

This section describes read-only hardware operations in a typical scenario.

Note: Read-only hardware functionality does not require WORM disk array hardware. You may remove write privilege from a raw device or file system file after the dbspace is altered read-only.

Creating an Archive

Follow these steps to create an archive.

Consider an IQ database consisting of a single catalog store dbspace named `db.db` with three main IQ dbspaces: `A`, `B` and `C`.

1. At time `t0`, alter all three main dbspaces read-only.
2. Copy `db.db` to `db.db0`, either by shutting the database down and copying `db.db` or using **dbbackup** to make a copy while the database is still running.

3. Freeze dbspaces `A`, `B` and `C` at the hardware level. Store `db.db0` in an immutable form, perhaps by storing it in a file system file on the WORM device and freezing it.

At this point, the database has been archived as of time `t0` in an immutable form.

See also

- *Creating More Archives* on page 450

Creating New Dbspaces

Follow these steps to create new dbspaces.

1. Create two new main dbspaces `D` and `E`.
2. Continue using the database `db.db` as a production database.

   The database objects (tables, indexes, etc.) that existed as of time `t0` may have changed so that `db.db` does not equal `db.db0`. `db.db` continues to read data from dbspaces `A`, `B` and `C` as long as the tables that existed at time `t0` exist and as long as they contain some unmodified rows of data that existed as of time `t0`. Even if or when this is no longer true, `db.db` will continue to open `A`, `B` and `C` unless they are dropped from `db.db`, which is only allowed if they are empty from `db.db`'s point of view.

Examining Archived Data

Follow these steps to examine archived data.

Prerequisites

Suppose that you need to examine the archived database as of time `t0`.

Task

1. Copy the archived read-only `db.db0` to a read-write file `db.db0.working`.
2. Start `db.db0.working`. Note that as long as the server name `db.db0.working` does not conflict with the production system `db.db`, there is no need to stop the production system. `db.db0.working` will open `A`, `B`, `C`, and `D` in read-only mode. This will not interfere with `db.db`'s use of these files on UNIX, although Windows returns a sharing violation.

   Note that the catalog file `db.db0.working` is open in read-write mode.
3. Create a user `inv` for an investigator who wishes to examine the archived database.
4. Grant `inv` `RESOURCE` permission to create views, stored procedures, global or local temporary tables or any other structures necessary for the investigation.

   `db.db0` as well as `A`,`B` and `C` remain unchanged.

Updating the Working Archive

Follow these steps to update the working archive.

Prerequisites

If years have passed since time `t0`, you may upgrade the `db.db0.working` as long as **ALTER DATABASE UPGRADE** modifies no objects in the IQ main store.

Task

1. The temporary dbspaces that existed as of time `t0` are not required to start `db.db0.working`. Use the server startup switch **-iqnotemp** to start `db.db0.working`
2. Drop and create new temporary dbspaces or use the temporary space created by the **-iqnotemp** parameter.

Creating More Archives

Follow these steps to create more archives.

Prerequisites

Create a new archive at time `t1` as follows.

Task

1. Alter dbspaces `D` and E read-only.
2. Copy db.db to `db.db1`.
3. Freeze `D` and E.
4. Save `db.db1` in an immutable form.
5. Create new main dbspace(s), e.g. `F` and `G`.
6. Continue to use the production system `db.db`.

At any time, it is possible to use the archived databases `db.db0` or `db.db1`, or even both simultaneously, by simply copying `db.db0` and/or `db.db1` to a working file and starting a server.

Create an archive and then follow this procedure to create any number of archived versions of `db.db`.

Creating More Archives

Follow these steps to create more archives.

See also

• *Creating an Archive* on page 448

System Recovery and Database Repair

Learn about normal Sybase IQ server recovery, special recovery modes, how to verify database consistency, and how to repair database inconsistencies.

When you restart the database server, Sybase IQ attempts to recover automatically. If the server cannot recover and restart, especially after a system failure or power outage, the database may be inconsistent.

See also

- *Data Protection* on page 405
- *Operating System Session Shutdown* on page 42
- *Resource Issues Running sp_iqcheckdb* on page 461
- *System Recovery* on page 385
- *Validating Your Database* on page 426

Recovery and Repair Overview

If your Sybase IQ server or database has problems restarting, use this information to diagnose database startup problems, verify the consistency of databases, and repair databases.

If you are able to restart the server after a failure, Sybase recommends that you verify your database, preferably before allowing users to connect. You verify databases using the **sp_iqcheckdb** stored procedure.

If you have trouble starting a server or database, if the database starts but users are unable to connect to it, or if problems are found during database verification, you may need to perform a forced recovery, restore the database, recover leaked space, or repair indexes.

Examining the Server Log and IQ Message Log

To determine what type of recovery or repair is needed, you need information from your server log (`servername.nnnn.srvlog`) and IQ message log (`dbname.iqmsg`). Be sure to retain this information so you can provide it to Sybase Technical Support if necessary.

For example, if data inconsistency is detected, the *dbname.iqmsg* file may include detailed diagnostic information.

Normal Recovery

During system recovery, any uncommitted transactions are rolled back and any disk space used for old versions (snapshots of database pages that were being used by transactions that did not commit) returns to the pool of available space.

After normal recovery, the database then contains only the most recently committed version of each permanent table, unless it is a multiplex database. A multiplex database contains all versions accessible to secondary servers.

During recovery from a system failure or normal system shutdown, Sybase IQ reopens all connections that were active. If the **-gm** option, which sets the number of user connections, was in effect at the time of the failure, you need to restart the IQ server with at least as many connections as were actually in use when the server stopped.

Database Verification

Use **sp_iqcheckdb** for database verification.

Check the consistency of your database as soon as possible after the server restarts following an abnormal termination, such as a power failure, and before performing a backup of the database.

You can use the **sp_iqcheckdb** stored procedure to detect and repair database consistency problems.

See also

- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

The sp_iqcheckdb Stored Procedure

The IQ Database Consistency Checker (DBCC) performs database verification. The **sp_iqcheckdb** stored procedure, in conjunction with server startup options, is the interface to DBCC.

You select the different modes of check and repair by specifying an **sp_iqcheckdb** command string. **sp_iqcheckdb** reads every database page and checks the consistency of the database, unless you specify otherwise in the command string.

Note: On a secondary server **sp_iqcheckdb** does not check the free list. It performs all other checks.

DBCC has three modes that perform increasing amounts of consistency checking and a mode for resetting allocation maps. Each mode checks all database objects, unless individual dbspaces, tables, partitions, indexes, or index types are specified in the **sp_iqcheckdb** command string. If you specify individual table names, all indexes within those tables are also checked.

Note: The **sp_iqcheckdb** stored procedure does not check referential integrity or repair referential integrity violations.

Refer to “sp_iqcheckdb procedure” in “System Procedures,” in *Reference: Building Blocks, Tables, and Procedures* for the complete syntax of **sp_iqcheckdb**.

DBCC Performance

The execution time of DBCC varies according to the size of the database for an entire database check, the number of tables or indexes specified, and the size of the machine. Checking only a subset of the database, i.e., only specified tables, indexes, or index types, requires less time than checking an entire database. Refer to Table 7-12 in *Reference: Building Blocks, Tables, and Procedures > System Procedures* for processing times of the **sp_iqcheckdb** modes.

For the best DBCC performance, be as specific as possible in the **sp_iqcheckdb** command string. Use the 'allocation' or 'check' verification mode when possible and specify the names of tables or indexes, if you know exactly which database objects require checking.

sp_iqcheckdb Check Mode

In check mode, **sp_iqcheckdb** performs an internal consistency check on all IQ indexes and checks that each database block has been allocated correctly. All available database statistics are reported. This mode reads all data pages and can detect all types of allocation problems and most types of index inconsistencies. Check mode should run considerably faster than verify mode for most databases.

When to run in check mode:

- If metadata, null count, or distinct count errors are returned when running a query.

Examples of check mode:

Table 45. sp_iqcheckdb Check Mode Examples

Command	Description
`sp_iqcheckdb 'check database'`	Internal checking of all tables and indexes in the database
`sp_iqcheckdb 'check table t1'`	Default checking of all indexes in table `t1`
`sp_iqcheckdb 'check index t1c1hg'`	Internal checking of index `t1c1hg`
`sp_iqcheckdb 'check indextype FP database'`	Checking of all indexes of type FP in the database

sp_iqcheckdb Verify Mode

In verify mode, **sp_iqcheckdb** performs an intra-index consistency check, in addition to internal index consistency and allocation checking. All available database statistics are reported. The contents of each non-FP index is verified against its corresponding FP index(es). Verify mode reads all data pages and can detect all types of allocation problems and all types of index inconsistencies.

When to run in verify mode:

- If metadata, null count, or distinct count errors are returned when running a query

Examples of verify mode:

Table 46. sp_iqcheckdb Verify Mode Examples

Command	Description
`sp_iqcheckdb 'verify database'`	Verify contents of all indexes in the database
`sp_iqcheckdb 'verify table t1'`	Verify contents of all indexes in table `t1`
`sp_iqcheckdb 'verify index t1c1hg'`	Verify contents of index `t1c1hg`
`sp_iqcheckdb 'verify indextype HG table t1'`	Verify contents of all HG indexes in table `t1`

Note: If you check individual non-FP indexes in check mode, the corresponding FP index(es) are automatically verified with internal consistency checks and appear in the DBCC results.

sp_iqcheckdb Allocation Mode

In allocation mode, **sp_iqcheckdb** checks that each database block is allocated correctly according to the internal physical page mapping structures (blockmaps). Database statistics pertaining to allocation are also reported. This mode executes very quickly. Allocation mode, however, does not check index consistency and cannot detect all types of allocation problems.

When to run in allocation mode:

- To check for leaked blocks or inconsistent indexes due to multiply owned blocks
- After forced recovery, run **sp_iqcheckdb** in **dropleaks** mode to reset the allocation map (must use database as the target)
- To check for duplicate or unowned blocks (use database or specific tables or indexes as the target)
- If you encounter page header errors

Examples of allocation mode:

Table 47. sp_iqcheckdb Allocation Mode Examples

Command	Description
`sp_iqcheckdb 'allocation database'`	Allocation checking of entire database
`sp_iqcheckdb 'allocation database dumpleaks'`	Allocation checking of entire database and print block numbers for leaked blocks to IQ message file
`sp_iqcheckdb 'allocation table t1'`	Allocation checking of table `t1`
`sp_iqcheckdb 'allocation index t1c1hg'`	Allocation checking of index `t1c1hg`
`sp_iqcheckdb 'allocation indextype LF table t2'`	Allocation checking of all LF indexes in table `t2`

If some partitions of the table are offline, you can specify a partition target to check only part of a table.

You can combine all modes and run multiple checks on a database in a single session. In the following example, **sp_iqcheckdb** performs a quick check of partition `p1` in table `t2`, a detailed check of index `i1`, and allocation checking for the entire database using half of the CPUs:

```
sp_iqcheckdb 'check table t2 partition p1
verify index i1
allocation database resources 50'
```

Allocation mode options are only allowed with the DBCC command 'allocation database'.

The following allocation mode options print block numbers for affected database blocks to the IQ message file:

- **dumpleaks** — leaked blocks
- **dumpdups** — duplicate blocks
- **dumpunallocs** — unallocated blocks

The **resetclocks** option corrects the values of internal database versioning clocks, in the event that these clocks are slow. Do not use the **resetclocks** option for any other purpose unless you contact Sybase IQ Technical Support.

The **resetclocks** option must be run in single user mode and is only allowed with the DBCC command 'allocation database'. The syntax of the **resetclocks** command is:

```
sp_iqcheckdb 'allocation database resetclocks'
```

sp_iqcheckdb Dropleaks Mode

When the Sybase IQ server runs in single-node mode, you can use dropleaks mode with either a database or dbspace target to reset the allocation map for the entire database or specified dbspace targets. If the target is a dbspace, then the dropleaks operation must also prevent read-write operations on the named dbspace. All dbspaces in the database or dbspace list must be online.

In the following example, the first statement resets allocation maps for the entire database, and the second statement resets allocation maps for the dbspace `dbsp1`.

```
sp_iqcheckdb 'dropleaks database'
sp_iqcheckdb 'dropleaks dbspace dbsp1'
```

Note: Use **sp_iqrebuildindex** to repair index errors. There is currently no support for repairing join indexes.

sp_iqcheckdb Output

The output of **sp_iqcheckdb** consists of an extensive list of statistics and any errors reported by DBCC.

Only non-zero values are displayed. Lines containing errors are flagged with asterisks (*****). Note that if you encounter errors, some of the statistics reported by DBCC may be inaccurate.

The output of **sp_iqcheckdb** is always copied to the IQ message file (`.iqmsg`). To redirect the **sp_iqcheckdb** output to a file, enter the following command:

```
sp_iqcheckdb ># file_name
```

where *file_name* is the name of the file to receive the output.

When the **DBCC_LOG_PROGRESS** option is ON, **sp_iqcheckdb** sends progress messages to the IQ message file. These messages allow the user to follow the progress of the **sp_iqcheckdb** procedure as it executes.

The following is sample progress log output of the command `sp_iqcheckdb 'check database'`

```
IQ Utility Check Database
Start CHECK STATISTICS table: tloansf
Start CHECK STATISTICS for field: aqsn_dt
Start CHECK STATISTICS processing index:
ASIQ_IDX_T444_C1_FP
```

```
Start CHECK STATISTICS processing index:
tloansf_aqsn_dt_HNG
Done CHECK STATISTICS field: aqsn_dt
```

Future Version Errors

If you see the message `DBCC Future Version Errors`, a DDL operation has been performed since the DBCC transaction began. DBCC continues to process the remaining tables, but leaked block checking is not performed and statistics do not include the tables that were skipped.

To avoid DBCC Future Version errors, execute the **COMMIT** command before you run **sp_iqcheckdb**.

The following DBCC output indicates a Future Version error:

```
=======================================|
===========================|=====
DBCC Verify Mode Report                |                           |
=======================================|
===========================|=====
** DBCC Future Version Errors          |1                          |
*****
```

Sample Output of Valid Database

The following is an example of running **sp_iqcheckdb** in verify mode. No errors are detected, there is no leaked space, the database allocation is consistent, and all indexes are consistent.

The command line for this example is **sp_iqcheckdb 'verify database'**. Note that DBCC verifies all indexes, but the index verification output shown here is abbreviated.

Each index that DBCC determines to be consistent is marked as verified in the result set.

```
        Stat                              Value                  Flags
=================================|
===================================|=====
DBCC Verify Mode Report          |                                 |
=================================|
===================================|=====
   DBCC Status                   |No Errors Detected                |
   DBCC Work units Dispatched    |75                               |
   DBCC Work units Completed     |75                               |
=================================|
===================================|=====
Index Summary                    |                                 |
=================================|
===================================|=====
   Verified Index Count          |86                               |
=================================|
===================================|=====
Allocation Summary               |                                 |
=================================|
===================================|=====
   Blocks Total                  |8192                             |
   Blocks in Current Version     |4855                             |
```

```
   Blocks in All Versions        |4855                            |
   Blocks in Use                 |4855                            |
   % Blocks in Use               |59                              |
=================================|
=================================|=====
Allocation Statistics            |                                |
=================================|
=================================|=====
   DB Extent Count               |1                               |
   Blocks Created in Current TXN |211                             |
   Blocks To Drop in Current TXN |212                             |
   Marked Logical Blocks         |8240                            |
   Marked Physical Blocks        |4855                            |
   Marked Pages                  |515                             |
   Blocks in Freelist            |126422                          |
   Imaginary Blocks              |121567                          |
   Highest PBN in Use            |5473                            |
   Total Free Blocks             |3337                            |
   Usable Free Blocks            |3223                            |
   % Total Space Fragmented      |1                               |
   % Free Space Fragmented       |3                               |
   Max Blocks Per Page           |16                              |
   1  Block Page Count           |104                             |
   3  Block Page Count           |153                             |
   ...
   16 Block Hole Count           |199                             |
=================================|
=================================|=====
Index Statistics                 |                                |
=================================|
=================================|=====
   ...
   Verified Index                |fin_data.DBA.ASIQ_IDX_T209_C3_HG  |
   Verified Index                |fin_data.DBA.ASIQ_IDX_T209_C4_FP  |
   Verified Index                |product.DBA.ASIQ_IDX_T210_C1_FP   |
   ...
   Verified Index                |employee.DBA.ASIQ_IDX_T212_C20_FP |
   Verified Index                |iq_dummy.DBA.ASIQ_IDX_T213_C1_FP  |
   FP Indexes Checked            |68                              |
   HNG Indexes Checked           |1                               |
   HG Indexes Checked            |17                              |
=================================|
=================================|=====
   ...
```

The DBCC output also contains extensive statistical information grouped under headings such as Container Statistics, Buffer Manager Statistics, catalog Statistics, Connection Statistics, and Compression Statistics. You can see an example of the available statistics by executing the command `sp_iqcheckdb 'verify database'` after connecting to the Sybase IQ demonstration database `iqdemo`.

See also

- *Analysis of Index Errors* on page 462
- *DBCC Error Messages* on page 476

- *Index Error Repair* on page 465
- *Dropping Inconsistent Indexes, Tables, or Columns* on page 475
- *Recovering Leaked Space* on page 472

Resource Issues Running sp_iqcheckdb

sp_iqcheckdb reports resource issues encountered while executing.

Messages describing resource issues are reported in the **sp_iqcheckdb** output or in the `.iqmsg` file.

- `Out of memory` and `DBCC Out of Memory Errors` You do not have enough memory for this operation. You may need to prevent other IQ operations or other applications from running concurrently with the **sp_iqcheckdb** stored procedure.
- `No buffers available` and `DBCC Out of Buffers Errors` The DBA may need to increase the buffer cache size.

Buffer cache sizes are set permanently using the database option `TEMP_CACHE_MEMORY_MB`. Use the server startup switches **-iqmc** and **-iqtc** to override the buffer cache size values set using the database options.

Do not run multiple database consistency checks at the same time, as DBCC is optimized to run one instance.

The CPU utilization of DBCC can be limited by specifying the **sp_iqcheckdb** parameter **resources resource-percent**, which controls the number of threads with respect to the number of CPUs. The default value of *resource-percent* is 100, which creates one thread per CPU and should match the load capacity of most machines. Set *resource-percent* to a value less than 100 to reduce the number of threads, if you are running DBCC as a background process. The minimum number of threads is 1.

If *resource-percent* > 100, then there are more threads than CPUs, which may increase performance for some machine configurations.

The database option **DBCC_PINNABLE_CACHE_PERCENT** can be used to tune DBCC buffer usage. The default of **DBCC_PINNABLE_CACHE_PERCENT** is to use 50% of cache. See *Reference: Statements and Options*.

See also

- *Data Protection* on page 405
- *Operating System Session Shutdown* on page 42
- *System Recovery* on page 385
- *System Recovery and Database Repair* on page 453
- *Validating Your Database* on page 426

Database Repair

Allocation problems can be repaired by running **sp_iqcheckdb** in **dropleaks** mode.

If DBCC detects index inconsistencies while attempting allocation repair, an error is generated and allocation problems are not fixed.

See also

- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

Analysis of Index Errors

Use **sp_iqcheckdb** to analyze index inconsistencies.

Sample of Output with Inconsistent Index

The following is an example of the type of output you see when you run **sp_iqcheckdb** and there is index inconsistency. DBCC displays both a summary and details about the indexes checked. The Index Summary section at the top of the report indicates if any inconsistent indexes were found. The names of the inconsistent indexes and the type(s) of problems can be found in the index statistics section. The lines with asterisks (*****) contain information about inconsistent indexes.

Extra, missing, or duplicate RID errors are the most common types of errors reported. These errors are an indication that the index is misrepresentative of the data and may give incorrect results or cause other failures. These errors are generally accompanied by other errors indicating the specifics of the inconsistencies.

In this example, DBCC reports an inconsistent HNG index. Because the corresponding FP index checks are good, the FP index can be used with **sp_iqrebuildindex** to repair the damaged HNG index.

The command line executed for this example is `sp_iqcheckdb 'verify database'`. Note that DBCC produces a detailed report, but some lines of the output have been removed in this example.

```
             Stat                              Value                   Flags
==============================|==================================|
=====
DBCC Verify Mode Report       |                                  |
==============================|==================================|
=====
** DBCC Status               |Errors Detected                  |*****
   DBCC Work units Dispatched |75                                |
   DBCC Work units Completed  |75                                |
==============================|==================================|
=====
Index Summary                 |                                  |
==============================|==================================|
=====
** Inconsistent Index Count   |1                                |*****
   Verified Index Count       |85                                |
==============================|==================================|
=====
Index Statistics              |                                  |
==============================|==================================|
=====
** Inconsistent Index        |contact.DBA.idx01_HNG            |*****
   ...
   Verified Index             |fin_data.DBA.ASIQ_IDX_T209_C3_HG  |
   Verified Index             |fin_data.DBA.ASIQ_IDX_T209_C4_FP  |
   ...
   Verified Index             |employee.DBA.ASIQ_IDX_T212_C19_FP |
   Verified Index             |employee.DBA.ASIQ_IDX_T212_C20_FP |
   Verified Index             |iq_dummy.DBA.ASIQ_IDX_T213_C1_FP  |
** Extra Index RIDs          |5                                 |*****
   FP Indexes Checked         |68                                |
   HNG Indexes Checked        |1                                 |
   HG Indexes Checked         |17                                |
                              |                                  |
```

The inconsistent index detected by **sp_iqcheckdb** is `contact.DBA.idx01_HNG`.

The following DBCC output is generated when **sp_iqcheckdb** is run again to check just the inconsistent index. The command line executed for this example is `sp_iqcheckdb 'verify index DBA.contact.idx01_HNG'`.

```
            Stat                                Value                  Flags
==============================|==================================|
=====
DBCC Verify Mode Report       |                                  |
==============================|==================================|
=====
** DBCC Status               |Errors Detected                  |*****
   DBCC Work units Dispatched |1                                 |
   DBCC Work units Completed  |1                                 |
                              |                                  |
==============================|==================================|
=====
Index Summary                 |                                  |
==============================|==================================|
```

```
=====
** Inconsistent Index Count   |1                              |*****
   Verified Index Count        |1                              |
                               |                               |
==============================|==================================|
=====
Index Statistics               |                               |
==============================|==================================|
=====
** Inconsistent Index        |contact.DBA.idx01_HNG           |*****
   Verified Index              |contact.DBA.ASIQ_IDX_T206_C1_FP   |
** Extra Index RIDs          |5                               |*****
   FP Indexes Checked          |1                              |
   HNG Indexes Checked         |1                              |
                               |                               |
==============================|==================================|
=====
```

DBCC Index Errors

The DBCC output contains messages related to problems with indexes.

Table 48. DBCC Index Errors

DBCC message	Description/action
Inconsistent Index Count	The number of indexes that DBCC found to have inconsistencies.
Inconsistent Index	The name of an index that DBCC found to be inconsistent.
Extra Index RIDs Missing Index RIDs Duplicate Index RIDs	The total number of rows that are inconsistent for all inconsistent indexes.
Bitmap Verify Errors	The total number of inconsistent bitmaps in all database objects
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.
Non-Completed Index Count	The number of indexes that could not be verified, because an exception occurred while checking.
Non-Completed Index	The name of an index that was not verified because an exception occurred while checking. If the exception is a future version, out of memory, or out of buffers error, commit the DBCC connection and re-run DBCC.

DBCC message	Description/action
VDO Incorrect First Available Fields VDO Incorrect Next Available Fields VDO Incorrect Used Count Fields VDO Incorrect In-use Bitvec VDO Incorrect In-use Bitmap VDO Incorrect Partial Bitmap VDO Incorrect Deleted Bitmaps	Unrepairable errors that can cause entire tables to be inaccessible. You must force drop the inconsistent table to resolve these errors.
HG Missing Groups HG Extra Groups HG Extra Keys HG Missing Keys B-Tree Invalid Item Count B-Tree Invalid Item Count G-Array Empty Page Errors G-Array Bad Group Type Errors G-Array Out of Order Group Errors	High Group index specific errors.

See also

- *DBCC Error Messages* on page 476
- *Index Error Repair* on page 465
- *sp_iqcheckdb Output* on page 458
- *Dropping Inconsistent Indexes, Tables, or Columns* on page 475
- *Recovering Leaked Space* on page 472
- *Index Problems that DBCC Cannot Repair* on page 475

Index Error Repair

Use the **sp_iqrebuildindex** procedure to repair an index, then run **sp_iqcheckdb** in verify mode to check for index inconsistencies.

If an index is still inconsistent, drop and recreate the index, and then rebuild the index.

Note: The **sp_iqrebuildindex** procedure cannot repair FP indexes. Sybase IQ has no functionality to repair FP indexes.

See also

- *Analysis of Index Errors* on page 462
- *DBCC Error Messages* on page 476
- *sp_iqcheckdb Output* on page 458
- *Dropping Inconsistent Indexes, Tables, or Columns* on page 475
- *Recovering Leaked Space* on page 472
- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

Analysis of Allocation Problems

Use **sp_iqcheckdb** to analyze allocation problems.

The database maintains an allocation map, also known as a free list, which tracks the blocks that are in use by database objects.

DBCC detects three types of allocation problems:

- Leaked blocks—A leaked block is a block that is allocated according to the database allocation map, but is found not to be part of any database objects. DBCC can recover leaked blocks.
- Unallocated blocks—An unallocated block is a block that is not allocated according to the database allocation map, but is found to be in use by a database object. DBCC can recover unallocated blocks.
- Multiply-owned blocks—A multiply-owned block is a block that is in use by more than one database object. At least one of the structures involved contains inconsistent data. DBCC cannot repair this type of allocation problem. If you encounter this type of error, run DBCC again, specifying a list of indexes, until you identify the indexes that share the block. These indexes must then all be dropped to eliminate the multiply-owned block.

Sample of Leaked Space Output

This is an example of the output you see when you run **sp_iqcheckdb** and there is leaked space. Lines with asterisks (*****) contain information about allocation problems. In this example, DBCC reports 16 leaked blocks.

The command line executed for this example is `sp_iqcheckdb 'allocation database'`.

```
         Stat                                    Value                    Flags
==============================|==============================|=====
DBCC Allocation Mode Report   |                              |
==============================|==============================|=====
** DBCC Status                |Errors Detected               |*****
   DBCC Work units Dispatched |164                           |
   DBCC Work units Completed  |164                           |
                              |                              |
==============================|==============================|=====
Allocation Summary            |                              |
==============================|==============================|=====
   Blocks Total               |8192                          |
   Blocks in Current Version  |4785                          |
   Blocks in All Versions     |4785                          |
   Blocks in Use              |4801                          |
   % Blocks in Use            |58                            |
** Blocks Leaked              |16                            |*****
                              |                              |
==============================|==============================|=====
Allocation Statistics         |                              |
==============================|==============================|=====
   ...
** 1st Unowned PBN            |1994                          |*****
   ...
==============================|==============================|=====
```

If one or more dbspaces are offline, use the following syntax to show allocation problems for a particular dbspace:

```
sp_iqcheckdb 'allocation dbspace dbspace-name'
```

DBCC Allocation Errors

Allocation problems are reported in the output generated by DBCC with **sp_iqcheckdb** run in a allocation mode or verification mode. If the Allocation Summary section has values flagged with asterisks, such as “** Blocks Leaked” or “** Blocks with Multiple Owners,” then there are allocation problems.

Messages in the DBCC output related to allocation problems are listed in the following table.

Table 49. DBCC Allocation Errors

DBCC message	Description/action
Block Count Mismatch	This count always accompanies other allocation errors.
Blocks Leaked 1st Unowned PBN	Blocks that were found not to be in use by any database object. Use **sp_iqcheckdb dropleaks** mode to repair.
Blocks with Multiple Owners 1st Multiple Owner PBN	Blocks in use by more than one database object. Drop the object that is reported as inconsistent.

DBCC message	Description/action
Unallocated Blocks in Use 1st Unallocated PBN	Blocks in use by a database object, but not marked as in use. Use **sp_iqcheckdb dropleaks** mode to repair.

If the Allocation Summary lines indicate no problem, but the Index Summary section reports a value for "Inconsistent Index Count," then this indicates one or more inconsistent indexes.

See also

- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

Repairing Allocation Problems using DBCC

Use **sp_iqcheckdb dropleaks** to repair database allocation problems.

Note: This procedure uses the **-gd** and **-gm** switches to restrict database access. For a more restrictive method, see *System Recovery and Database Repair > Database Repair > Forced Recovery Mode > Starting a Server in Forced Recovery Mode*.

1. Start the server.

   For example:

   ```
   start_iq -n my_db_server -x 'tcpip{port=7934}'
   -gd dba -gm 1 /work/database/my_db.db
   ```

   Note: You must start the database with the ".db" extension, not ".DB".

   Sybase recommends using two server startup switches to restrict access:

   - Use **-gd DBA** so that only users with DBA authority can start and stop databases. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
   - Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

For more information about restricting connections, see *Starting a Server in Forced Recovery Mode*.

2. Run the stored procedure **sp_iqcheckdb** in dropleaks mode:

   ```
   sp_iqcheckdb 'dropleaks database'
   ```

   If one or more dbspaces are offline, you can repair allocation problems for a dbspace alone by running:

   ```
   sp_iqcheckdb 'dropleaks dbspace dbspace-name'
   ```

   If the allocation repair is successful, **sp_iqcheckdb** displays the message “Freelist Updated.” If errors are detected, **sp_iqcheckdb** returns the messages “Freelist Not Updated” and “Errors Detected.”

3. Stop the server after **sp_iqcheckdb** finishes. To stop the server, use **stop_iq** on UNIX or the shutdown button in the console window on Windows.

After allocation problems are repaired, allocation statistics appear in the DBCC output with no errors.

DBCC displays an Allocation Summary section at the top of the report, which lists information about allocation usage. The Allocation Statistics section provides more details about the blocks. The DBCC output does not contain repair messages for the leaked blocks that have been recovered.

For example:

```
sp_iqcheckdb 'dropleaks dbspace mydbspace';
checkpoint;
```

The **sp_iqcheckdb** output indicates no errors, so the **checkpoint** is executed.

DBCC reports statistics that do not show in this abbreviated output.

```
         Stat                              Value                       Flags
==============================|====================================|
=====
DBCC Allocation Mode Report   |                                    |
==============================|====================================|
=====
   DBCC Status                |Freelist Updated                    |
   DBCC Status                |No Errors Detected                  |
   DBCC Work units Dispatched |75                                  |
   DBCC Work units Completed  |75                                  |
==============================|====================================|
=====
Allocation Summary            |                                    |
==============================|====================================|
=====
   Blocks Total               |8192                                |
   Blocks in Current Version  |4594                                |
   Blocks in All Versions     |4594                                |
   Blocks in Use              |4610                                |
```

```
   % Blocks in Use            |56                                  |
==============================|====================================|
=====
Allocation Statistics         |                                    |
==============================|====================================|
=====
   DB Extent Count            |1                                   |
   Marked Logical Blocks      |8176                                |
   Marked Physical Blocks     |4594                                |
   Marked Pages               |511                                 |
   Blocks in Freelist         |126177                              |
   Imaginary Blocks           |121567                              |
   Highest PBN in Use         |5425                                |
   Total Free Blocks          |3582                                |
   Usable Free Blocks         |3507                                |
   % Free Space Fragmented    |2                                   |
   Max Blocks Per Page        |16                                  |
   1  Block Page Count        |103                                 |
   3  Block Page Count        |153                                 |
   ...
   16 Block Hole Count        |213                                 |
==============================|====================================|
=====
```

Note: When performing forced recovery or leaked blocks recovery, you must start the database with the “.db” extension, not “.DB”. For example:

```
 start_iq -n my_db_server -x 'tcpip{port=7934}'
-gd dba my_db /work/database/my_db.db
```

Forced Recovery Mode

Forced database recovery differs from normal database recovery.

- **Forced recovery marks all storage within the database as in use –** In order to recover a potentially inconsistent allocation map, all storage within the database is marked as in use. Use the **sp_iqcheckdb** in **dropleaks** mode to reset the allocation map to the correct state.
- **Incremental backups are disabled –** After the database is opened in forced recovery mode, incremental backups are disabled. The next backup must be a full backup. Doing a full backup reenables incremental backups.
- **The forced recovery parameter applies to all opens of the database while the server is up –** Therefore, after the database is opened, the DBA needs to bring the server back down, and then restart the server without the forced recovery flag, to be sure that subsequent opens run in regular mode. Repeated opens of the database with forced recovery on do not harm the database, but could be confusing to the DBA. Each time you open the database in forced recovery mode, all the storage within the database is marked as in use.

Starting a Server in Forced Recovery Mode

Forced recovery allows the server to start if the allocation map is inconsistent.

In forced recovery mode, options display information about inconsistencies. You can also specify options to repair such inconsistencies.

Follow this procedure only if you see s_buf or free list errors during failure recovery. If SQL Anywhere recovery errors occur, **-iqfrec** will not correct the problem.

Restricting database access gives the DBA greater control over inadvertent opens of the database during forced recovery. Sybase recommends using two server startup switches to restrict access:

- Use **-gd DBA** so that only users with DBA authority can start and stop databases on a running server. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
- Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

An alternate way to restrict connections is to specify

```
sa_server_option('disable_connections', 'ON')
```

just after you start the connection where you are performing forced recovery and

```
sa_server_option('disable_connections', 'OFF')
```

on the same connection after recovery. The disadvantage is that this method precludes emergency access from another DBA connection.

1. Start the server with forced recovery (to mark all pages as used), using the **-iqfrec** server startup option in the **start_iq** command. For example:

   ```
   start_iq -n my_server -x 'tcpip(port=7934}'
   -gd dba -gm 1 -iqfrec my_db /database/my_db.db
   ```

   Forced recovery starts the server in single-node mode. Stop all secondary servers first.

   Warning! You must specify the override startup switch (**-iqmpx_ov 1**) and start in single node mode (**-iqmpx_sn 1**) when starting a multiplex write server after any failure. Never use multiplex mode (the default) for recovery.

   You specify the database name twice, once to specify the database undergoing forced recovery and once to specify the database to start. The **-iqfrec** server startup option requires the database name. Note that this is the physical database name, which is case sensitive. Do not use **select_dbname** to determine the database name, as it returns the logical name assigned by the **-n** startup option.
2. If desired, you can run **sp_iqcheckdb** to check for leaked blocks.

3. Stop the server after it has started successfully. To stop the server, use **stop_iq** on UNIX or the shutdown button in the console window on Windows.
4. Restart the server using your usual method, without the **-iqfrec** option.

If you are unable to start your server in forced recovery mode, contact Sybase Technical Support.

Using Forced Recovery without a Follow On sp_iqcheckdb

Running forced recovery starts the database in a valid, but fully allocated mode. In other words, you should be able to do all operations, but no permanent main dbspace is left. Before you do anything else, you must either recover the lost dbspace by running **sp_iqcheckdb** in **dropleaks** mode, or add a new dbspace. Note that queries should also run successfully, since they do not need additional permanent dbspace; however, you cannot load, insert, or delete data.

Warning! Running queries without verifying the database will not cause any inconsistency in your data. However, if there is a problem in the data that caused the server to fail, the server could fail again or produce incorrect results.

See also

- *Recovering Leaked Space* on page 472
- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520

Recovering Leaked Space

Use the **sp_iqcheckdb** stored procedure in **dropleaks** mode to recover leaked storage space within the specified database.

An allocation map is used by the server to determine if a page is in use or not in use within IQ. Either through system failure or as a result of opening a database with forced recovery, the allocation map of the database may not reflect the true allocation of its usage. When this occurs, we say that the database has “leaked” storage or “leaked blocks.” In general, you need not be concerned about small numbers of leaked blocks. If you have many megabytes of leaked blocks, you probably want to recover that space.

When leaked storage is being recovered, other transactions that alter the allocation map are shut out. Such operations include checkpoints and commands that modify the database.

You can recover leaked storage and force recovery either at the same time or separately. To recover leaked space within a database without doing a forced recovery, repair allocation problems using DBCC. To recover leaked space within a database after doing a forced recovery, recover leaked space using this procedure.

If repairing allocation problems using DBCC fails to recover leaked storage, then use this procedure.

Note: This procedure uses the **-gd** and **-gm** switches to restrict database access. For a more restrictive method, start the server in forced recovery mode.

1. Start the server with the `-iqfrec` option in the **start_iq** command. For example:

   ```
   start_iq -n my_db_server -x 'tcpip{port=7934}'
   -gd dba -gm 1
   -iqfrec my_db /work/database/my_db.db
   ```

   You specify the database name twice in a row, once to specify it as the database you are starting, and once to specify it as the database undergoing forced recovery. The **-iqfrec** option requires the database name.
2. Connect to the database you are recovering.
3.

   Run the stored procedure **sp_iqcheckdb** in dropleaks mode:

   ```
   sp_iqcheckdb 'dropleaks database'
   ```

   If there are no errors and **sp_iqcheckdb** displays the message `Freelist Updated`, you have recovered leaked space and forced recovery. Continue to the next step.

   If inconsistency is found, drop inconsistent indexes, tables, or columns. Then run **sp_iqcheckdb** again to recover leaked space.
4. Issue a checkpoint.
5. Stop the server using your usual method.
6. Restart the server using your usual method, and proceed with normal processing.

See also

- *Analysis of Index Errors* on page 462
- *DBCC Error Messages* on page 476
- *Index Error Repair* on page 465
- *sp_iqcheckdb Output* on page 458
- *Starting a Server in Forced Recovery Mode* on page 471
- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497

- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520

Recovering Multiplex Databases

Before troubleshooting recovery problems with a multiplex database, see Using Sybase IQ Multiplex.

Problems Reported by DBCC

Messages are reported for problems that DBCC cannot repair.

Table 50. Messages for Problems DBCC Cannot Repair

DBCC message	Description/action
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.
VDO Incorrect First Available Fields VDO Incorrect Next Available Fields VDO Incorrect Used Count Fields VDO Incorrect In-use Bitvec VDO Incorrect In-use Bitmap VDO Incorrect Partial Bitmap VDO Incorrect Deleted Bitmaps	Unrepairable errors that can cause entire tables to be inaccessible. You must force drop the inconsistent table to resolve these errors.
Blocks with Multiple Owners 1st Multiple Owner PBN	Blocks in use by more than one database object. Drop the object that is reported as inconsistent.
DBCC Meta-data Errors Blockmap Invalid Chunksize Error Count Blockmap Compression Bit Error Count Blockmap Invalid Block Number Error Count	An internal page mapping structure is inconsistent and the object needs to be dropped.
DBCC Inconsistent Disk Block Headers DBCC Decompress Errors	The storage for the object is inconsistent and the object needs to be dropped.

See also

- *DBCC Error Messages* on page 476

Index Problems that DBCC Cannot Repair

Use these suggestions to repair inconsistent indexes.

If DBCC detects a problem with an index, the name of the index is reported with the type of problem. Use **sp_iqrebuildindex** to repair a non-FP index. FP indexes cannot be repaired. Analyze index errors for indexes reported as “Inconsistent Index,” when **sp_iqcheckdb** is run in default or check mode.

Depending on the type of problem, use **DROP INDEX**, **ALTER TABLE DROP COLUMN**, **DROP TABLE**, or the **FORCE_DROP** option to resolve the problem.

Sybase recommends calling Sybase Technical Support for help in determining the best course of action to fix an inconsistent index or table.

See also

- *Analysis of Index Errors* on page 462

Dropping Inconsistent Indexes, Tables, or Columns

Use these suggestions to resolve issues with unrepairable indexes, columns, or tables.

If **sp_iqcheckdb** reports unrepairable indexes, columns, or tables, then these objects must be dropped using the **DROP INDEX**, **ALTER TABLE DROP COLUMN**, or **DROP TABLE** statements respectively.

Note: You should not attempt to force drop objects unless Sybase Technical Support has instructed you to do so.

If you cannot drop an inconsistent object, set the temporary `FORCE_DROP` option. `FORCE_DROP` causes the IQ server to silently leak the on-disk storage of the dropped object, rather than try to reclaim it. You can recover the leaked space later using DBCC. This is desirable for an inconsistent object, because the only information about the storage of an object is within the object itself, and this information is suspect for an inconsistent object.

The `FORCE_DROP` database option is not allowed on a secondary node. If a force drop is attempted on a secondary node, an error is returned. `FORCE_DROP` is a temporary option, so that the value of the option does not get propagated to secondary nodes at synchronization.

Note: When force dropping objects, you must ensure that only the DBA is connected to the database. Restart the server immediately after a force drop.

The following procedure uses the **-gd** and **-gm** switches to restrict database access. The **-gd** switch only limits users who can start or stop databases on a running server. For a more restrictive method, start the server in forced recovery mode.

1. Restart the server.

```
start_iq -n bad_db_server -x 'tcpip{port=7934}'
-gm 1 -gd dba bad_db.db
```

You must not allow other users to connect when force dropping objects.

Sybase recommends using two server startup switches to restrict access:

- Use **-gd DBA** so that only users with DBA authority can start and stop databases. (Note that the client must already have a connection to the server to start or stop the database, so this switch does not prevent connections.)
- Use **-gm 1** to allow a single connection plus one DBA connection above the limit so that a DBA can connect and drop others in an emergency.

For more information about restricting connections, see *Installation and Configuration Guide*.

2. Set the temporary option `FORCE_DROP` to ON.

```
set temporary option FORCE_DROP = 'ON'
```

3. Drop all inconsistent objects.

   Use the commands **DROP INDEX**, **ALTER TABLE DROP COLUMN**, or **DROP TABLE** as needed. Do not enter any other DDL or DML commands until after restarting the server.

4. Restart the server.

   To recover the leaked space and update the allocation map to the correct state, start the server.

```
start_iq -n bad_db_server -x 'tcpip{port=7934}'
-gm 1 -gd dba bad_db.db
```

5. Run sp_iqcheckdb.

```
sp_iqcheckdb 'dropleaks database';
```

   This step resets the database allocation map to the calculated allocation map.

See also

- *Analysis of Index Errors* on page 462
- *DBCC Error Messages* on page 476
- *Index Error Repair* on page 465
- *sp_iqcheckdb Output* on page 458

DBCC Error Messages

These are the most important messages in the DBCC output.

Table 51. DBCC Error Messages

DBCC message	Description/action
Inconsistent Index Count	The number of indexes that DBCC found to have inconsistencies.

DBCC message	Description/action
Inconsistent Index	The name of an index that DBCC found to be inconsistent.
Extra Index RIDs Missing Index RIDs Duplicate Index RIDs	The total number of rows that are inconsistent for all inconsistent indexes.
Bitmap Verify Errors	The total number of inconsistent bitmaps in all database objects.
FP Lookup Table Inconsistencies	An unrepairable error, where the 1-byte or 2-byte FP is internally inconsistent.
Non-Completed Index Count	The number of indexes that could not be verified, because an exception occurred while checking.
Non-Completed Index	The name of an index that was not verified because an exception occurred while checking. If the exception is a future version, out of memory, or out of buffers error, commit the DBCC connection and re-run DBCC.
HG Missing Groups HG Extra Groups HG Extra Keys HG Missing Keys B-Tree Invalid Item Count B-Tree Invalid Item Count G-Array Empty Page Errors G-Array Bad Group Type Errors G-Array Out of Order Group Errors	High Group index specific errors.
VDO Incorrect First Available Fields VDO Incorrect Next Available Fields VDO Incorrect Used Count Fields VDO Incorrect In-use Bitvec VDO Incorrect In-use Bitmap VDO Incorrect Partial Bitmap VDO Incorrect Deleted Bitmaps	Unrepairable errors that can cause entire tables to be inaccessible. You must force drop the inconsistent table to resolve these errors.
Block Count Mismatch	This count accompanies other allocation errors.
Blocks Leaked 1st Unowned PBN	Blocks that were found not to be in use by any database object. Use dropleaks mode to repair.
Blocks with Multiple Owners 1st Multiple Owner PBN	Blocks in use by more than one database object. Drop the object that is reported as inconsistent.
Unallocated Blocks in Use 1st Unallocated PBN	Blocks in use by a database object, but not marked as in use. Use dropleaks mode to repair.
Freelist Updated	Indicates successful allocation repair.
Freelist Not Updated	Indicates errors detected during allocation repair and the allocation repair was not successful.

DBCC message	Description/action
Invalid Blockmap Unique ID Generator Blockmap Unique ID Generator Updated Invalid Transaction ID Counter Transaction ID Generator Updated	Errors and repair messages specific to the DBCC **reset-clocks** option.
DBCC Future Version Errors	DBCC could not open the table, because DDL was performed on it. Commit the DBCC connection and re-run DBCC.
DBCC Locked Table Access Conflict	DBCC tried to open a table that another connection has locked. To ensure complete DBCC processing, make sure that no other users have locked tables in the database.
DBCC Out of Buffers Errors	The size of the IQ main cache is too small. Either increase the main cache size or run DBCC on individual objects.
DBCC Out of Memory Errors	There is insufficient system memory to complete the DBCC operation.
DBCC Meta-data Errors Blockmap Invalid Chunksize Error Count Blockmap Compression Bit Error Count Blockmap Invalid Block Number Error Count	An internal page mapping structure is inconsistent and the object needs to be dropped.
DBCC Page Read Errors	An I/O error occurred while trying to read an object. Perform hardware diagnostics.
DBCC Inconsistent Disk Block Headers DBCC Decompress Errors	The storage for the object is inconsistent and the object needs to be dropped.
DBCC Unknown Exceptions	An exception of a type unknown to DBCC occurred. Check the IQ message file for details.
Unowned LVC cells Duplicate LVC cell rows Unallocated LVC cell rows	Messages indicate inconsistencies with a `VARCHAR` or `CLOB` column. Unowned LVC cells represent a small amount of unusable disk space and can safely be ignored. Duplicate and Unallocated LVC cells are serious errors that can only be resolved by dropping the damaged columns. To drop a damaged column, create a new column from a copy of the old column, then drop the original column and alter rename the new column to the old column. LVC is a `VARCHAR` column with a width greater than 255. `CLOB` also uses LVC.

See also

- *Analysis of Index Errors* on page 462
- *Index Error Repair* on page 465
- *sp_iqcheckdb Output* on page 458

- *Dropping Inconsistent Indexes, Tables, or Columns* on page 475
- *Recovering Leaked Space* on page 472
- *Problems Reported by DBCC* on page 474

Troubleshooting Hints

Sybase IQ provides many resources for addressing any problems you encounter.

For information on resolving issues related specifically to Sybase IQ multiplex servers, see *Using Sybase IQ Multiplex*.

Sources of Online Support

If you are unable to resolve a problem using documentation, see the Sybase 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.

MySybase can be used 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 Sybase Web sites, see the *Release Bulletin*.

Solutions for Specific Conditions

You may need more information to diagnose and resolve certain issues.

You can use diagnostic tools to diagnose various conditions, including those described in the following sections.

Decision Flow for Server Recovery and Database Repair

You may experience trouble starting a server or database, if the database starts but you are unable to connect to it, or if problems are found during database verification.

1. Does the server start?

   If the server starts, go to step 2.

   If the server does not start, refer to the section *Server operational issues*. If you cannot start the server after following the suggestions in this section, then refer to the section *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. A restore of the database from backup may be necessary.
2. Can you connect to the database?

If you cannot connect to the database, refer to the section *Database connection issues* for troubleshooting suggestions.

If you can connect to the database and you previously started the server with forced recovery, refer to the section *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, refer to the section *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, you should run DBCC checks to validate the database. Refer to the section *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, refer to the section *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, refer to the section *Repairing Allocation Problems using DBCC* for the procedure to repair allocation problems using DBCC.

See also

- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520

Server Operational Issues

Issues may affect the operation of the server, including startup, shutdown, unresponsiveness, and abnormal termination.

See also

- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492

- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

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, Sybase IQ writes the error to the first one of the following that is defined:

- `$IQDIR15/logfiles/<servername>.nnnn.stderr`
- `$IQDIR15/logfiles/<servername>.nnnn.srvlog`
- The Systems applications log file

Possible causes include:

Transaction Log File Does Not Match the Database

The following 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, check to be sure 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, then 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.

The following messages appear:

```
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, then 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 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, the following 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 will not start, try to start the server using the **iqsrv15** command. The **iqsrv15** 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.

See also

- *You Cannot Run start_iq* on page 486

Log File Has Illegal Name

If you specified a separate request-level logging file, but the filename 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 a Sybase IQ server is running and you attempt to start another Sybase IQ server on the same machine using the same port number, messages appear in the server log file (`*.srvlog`).

These messages appear:

```
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 will not start, run the **stop_iq** command (UNIX) to display the names and port numbers of 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), Sybase IQ generates an available port number.

Here are the messages you see in the server log file, when you start the server and do not specify 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 Was 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 with the following 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-15_3/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**.

Not Enough Memory on Windows

If the Sybase IQ server will not start on a 32-bit Windows system, make sure you have enabled the Microsoft 4GT RAM Tuning feature, if appropriate for your version of Windows server.

The 4GT option configures the Windows operating system at boot time to allow the allocation of up to 3GB of dynamic memory for a user process. See “System requirements” in the chapter “Installing Sybase IQ” in the *Installation and Configuration Guide for Windows* for a list of supported Windows platforms and details on enabling the 4GT feature.

Environment Variables Are 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.

See *Reference: Building Blocks, Tables, and Procedures > File Locations and Installation Settings > Running UNIX environment source files* .

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 **Tools > Sybase IQ 15 > Start Server** in Sybase Central or the **iqsrv15** command.

Note: Use **iqsrv15** only for troubleshooting server start-up errors. Always use **start_iq** to start Sybase IQ servers.

Before running **iqsrv15**, 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: Be aware that **unlimit** affects soft limits only. You must change any hard limits by setting kernel parameters.

- Set all server options appropriately for your platform. For details about appropriate options and how to set them in a configuration file, see the *Installation and Configuration Guide*.
- Add the path `$IQDIR/lib64/ocs` to the environment to load the engine and required libraries before you invoke **iqsrv15**. Put this path in the environment only during testing, as follows:

  On AIX:

  ```
  % setenv LIBPATH "${LIBPATH}:{IQDIR15}/lib64/ocs"
  ```

  On other UNIX/LINUX platforms:

  ```
  % setenv LD_LIBRARY_PATH "${LD_LIBRARY_PATH}:${IQDIR15}/lib64/
  ocs"
  ```

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, Sybase IQ cannot find those database files.

Any server startup scripts should change directory to a known location before issuing the server startup command.

Syntax for **iqsrv15** is as follows:

```
iqsrv15 -n server-name -gm number
[ other-server-switches ] [ database-file [ database-switches ] ]
```

Note: On the **iqsrv15** command line, the last option specified takes precedence, so if you want to override your configuration file, list any options you want to change after the configuration file name. For example:

```
iqsrv15 @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 **iqsrv15** command, then attempt to start again using the **iqsrv15** utility with minimal switches and parameters. For example:

```
iqsrv15 -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 **iqsrv15** 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.

See also

- *Server Name is Not Unique on Your Network* on page 484

Sybase IQ Stops Processing or Stops Responding

You can detect the cause of server unresponsiveness by looking in the Sybase IQ message file.

Possible Causes

The following are the two most common causes of server unresponsiveness:

- 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 Sybase IQ database option QUERY_PLAN = 'ON' to log useful information in the 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 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 these options can be set higher for a large database.
- Determine if the Sybase IQ server process (`iqsrv15`) 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 Sybase IQ server process should be processing normally.

  If the Sybase IQ server CPU usage is normal, you can examine what the server is doing, i.e., what statement the server is currently executing.
- If there are no out of space indications, use **dbisql** on a new or existing connection to gather the following information, in the specified order.

Table 52. 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

Command	Informational purpose
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 Sybase Technical Support for assistance. They can use the information you have just gathered to help diagnose the problem.

- When the server is unresponsive, you can generate a stack trace for each Sybase IQ thread by creating a file named `DumpAllThreads` or `dumpallthreads` in the `$IQDIR15/logfiles` directory (the `%ALLUSERSPROFILE\%\SybaseIQ\logfiles` folder on Windows 32 and 64 platforms, `C:\ProgramData\SybaseIQ\logfiles` for Vista 64).

  Starting Sybase IQ as recommended, using the Program Manager or **start_iq** command, sets the *IQDIR15* variable automatically. If the *IQDIR15* variable is not set, create the `DumpAllThreads` file in the directory in which `iqsrv15` was started.

  The 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 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 10 minutes with a 10 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 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 to avoid a deadlock due to resource depletion is one or a combination of the following:

- Restrict the number of users on the server by reducing the value of the **-gm** server startup option
- Add another secondary server to a multiplex
- Increase the processing capacity of the hardware by adding CPUs

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Finding the Currently Executing Statement* on page 521
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *Logging Server Requests* on page 522

System Failure/Sybase IQ Failure

You can detect the cause of system/Sybase IQ failure by looking in the 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 will not be lost.
- On UNIX, send a copy of the stack trace to 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 Sybase IQ message log (default name `dbname.iqmsg`). This information helps 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 are able to start the server but not the database, check that database parameters are specified correctly on the startup line and/or 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 Sybase Technical Support. On Sun Solaris two different **ps** options are available. Use both.
   ```
   ps -aAdeflcj|egrep "PPID|iqsrv15"
   ```
   ```
   /usr/ucb/ps -awwwlx|egrep "PPID|iqsrv15"
   ```
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 are able to 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, and then reboot 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 Sybase 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 iqsrv15.exe and then click the **End Process** button to stop the database server.
3. If necessary, restart Windows.

See also

- *Adding Space if you Cannot Connect to a Server* on page 498

Database Connection Issues

You may encounter issues when attempting to connect to a database.

See also

- *Analysis of Allocation Problems* on page 466
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481
- *Reconnecting After You Restore* on page 436

Cannot Connect to a Database

You may experience problems connecting to a database.

Possible Causes

- Data source is not defined, or you have entered or defined it incorrectly.

  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. The error messages returned are:

  ```
  Unable to connect
  ```

  or,

  ```
  Could not connect to the database.
  ```

  followed by the message:

  ```
  Invalid user ID or password.
  ```

  Try connecting again with the correct user ID and password.
- 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. The error messages returned are:

  ```
  Unable to connect
  ```

or,

```
Could not connect to the database.
```

followed by the message:

```
Specified database not found.
```

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. The error messages returned are:

```
Unable to connect
Database server connection limit exceeded.
```

- You have run out of disk space. Check the Sybase IQ message file for messages related to disk space.
- The server name specified is not correct. The error messages returned are:

```
Connection failed.
Database server not running.
```

  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 the command **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 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. If the server does not support the requested character set, the connection fails.

  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.

Note: Do not confuse an inability to connect to a database with a Sybase IQ server-level error while Sybase IQ is trying to open a database.

Action

If you suspect that you cannot connect because there is a problem with the database, you can look in the `dbname.iqmsg` file to determine where the problem occurred.

If the message `Open Database Completed` appears, then 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.

See also

- *Required Command Line Options* on page 23

Interactive SQL (dbisql) Issues

This section contains information on troubleshooting issues related to the operation of **dbisql**.

Data Truncation or Data Conversion Error

Certain issues may cause data truncation or conversion errors.

Possible Causes

A data truncation error or conversion error occurs when a procedure calls another procedure with a dynamic result set and all of the following are true:

- The Sybase IQ server is version 12.5
- **dbisql** connects through iAnywhere JDBC driver
- **dbisql** version is higher than 7.04.

The problem does not happen if **dbisql** connects through the ODBC driver or if Sybase IQ 12.6 is used with **dbisql** 9.0.1.

- Differences in display characteristics between your terminal and the expectations of Sybase IQ.
- Differences in function key support between your terminal and the expectations of Sybase IQ.

Actions

- Connect **dbisql** through the ODBC driver.
- Use:
  - Sybase IQ 15.1 or 15.2 with **dbisql** version 11.0.1,
  - Sybase IQ 12.7 with **dbisql** version 9.0.2,
  - Sybase IQ 12.6 with **dbisql** version 9.0.1, or
  - Sybase IQ 12.5 with **dbisql** version 7.0.4
- Add a statement like the following to the start of the procedure, to keep the server from adding a result set:

```
IF 1 = 0 THEN
SELECT 1 AS a FROM nosuchtable;
END IF;
```

Directories Remain After Exiting dbisql

This issue affects users of NFS file systems only.

Possible Causes

- The IQTMP15 environment variable is not set to point to a local directory.

  Each client connection creates several directories and files in a temporary directory. Sybase IQ deletes these files when the connection ends. If IQTMP15 does not point to a local directory, it cannot find the `.nfs*` files that NFS creates.

Action

- Set IQTMP15 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.

See also

- *Setting Database Options in Sybase Central* on page 157

Insufficient Disk Space

The Sybase IQ server does not wait for additional space on an out-of-dbspace condition, but instead 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, then 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 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. For this reason, recognizing when you are low on disk space and adding a new dbspace before you run out of space are important.

Actions

- Check recent messages in the Sybase IQ message log (`dbname.iqmsg`). If you see an `out of space` message, you must add another dbspace. The message in the 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 Sybase IQ message log for the following messages:

  - If a buffer or dbspace allocation request fails because there is no space in the dbspace, the following 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.

    This error messages replaces the error message `You have run out of { IQ STORE | IQ TEMPORARY STORE } dbspace in database <`*`dbname`*`>. In another session, please issue a CREATE DBSPACE ... { IQ STORE | IQ TEMPORARY STORE } command and add a dbspace of at least `*`nn`*` MB.`

  - If the entire transaction is rolled back on an out-of-dbspace condition, the following error message is reported:

    ```
    %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, the following error message is returned 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, you know that 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 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 Sybase IQ is probably not operational. If the CPU usage does change, Sybase IQ is operational.
- Check the **sp_iqstatus** output for the following two lines:

  ```
  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, you need to 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, so 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.

See also

- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488

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. For multiplex servers, see *Using Sybase IQ Multiplex*.

See also

- *Analysis of Allocation Problems* on page 466
- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Server Operational Issues* on page 482

- *The sp_iqcheckdb Stored Procedure* on page 520
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

Effect of Checkpoints on Out of Disk Space Conditions

If 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 are unable to add space because you cannot connect to the server, add space using the **CREATE DBSPACE** command.

1. Shut down the database 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 Sybase IQ window, and 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. Rerun the operation that originally failed due to insufficient space.

See also

- *Server Fails to Shut Down* on page 491

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`), Sybase IQ returns the error `A dbspace has reached its maximum file size`. To prevent this problem:

- Monitor space usage periodically.
- 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 Sybase IQ database directory. The `.tmp` file is located under `$IQTMP15/<servername>/tmp`, or if $IQTMP15 is not set, under `/tmp/.SQLAnywhere/<servername>/tmp`.

Adding the Wrong Type of Space

If the temporary dbspace runs out of space and you accidentally omit the **TEMPORARY** keyword in the **CREATE DBSPACE** command, you cannot create a temporary dbspace.

Instead, add the file in the existing temporary dbspace as IQ_SYSTEM_TEMP.

Fragmentation

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 Main IQ Blocks Used is less than 100%, it usually indicates that your database is fragmented,

Freeing Space

Note that when a connection is out of space, freeing space by dropping tables or indexes in another connection is not possible, because the out of space transaction will see those objects in its snapshot version.

Reserving Space for the Future

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**. Set these options large enough to handle running out of space during a COMMIT or CHECKPOINT. See *Reference: Statements and Options* .

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 the event handler after the load completes.

The following is sample event handler code. You can modify this 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 iqsrv15 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 non-privileged operations. 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;
```

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488

Insufficient Threads

The required number of server threads may not be available for your query.

Possible Cause

If the client receives a message like `Not enough server threads available for this query [-1010011] ['QXA11']`, the query requires more 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 the system stored procedure **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. Remember that some threads may be assigned but idle.

- If the condition persists, you may need to restart the server and specify more IQ threads. Use the **-iqmt** server startup switch to increase the number of processing threads that Sybase IQ can use. The default is 60 threads per CPU for the first four CPUs and 50 threads per CPU for the remainder, 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 + 3 = 653`. 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, Sybase IQ may return the error `Ran out of threads. Start up server with more threads.` (SQLCODE -1012024). The **RESTORE** command will try to allocate a "team" of threads for the restore operation. Sybase IQ will try to allocate at least one thread per backup device plus two threads per CPU, plus one thread to the "team" for the restore. Make sure that enough threads have been allocated on 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.

If you see the error `AbortIfEndofStack` in the stack trace file (`stktrc-YYYYMMDD-HHNNSS_#.iq`), the thread stack has overflowed.

Possible Causes

- To avoid this problem, restart 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 not adequate, 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 Sybase Technical Support.

Unused Semaphores and Shared Memory Left After Abnormal Exit

Abnormal exits may leave unused semaphores and shared memory.

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, you should 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 Sybase IQ instances are started by the same user (as

Sybase recommends), 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 Procedure Identifiers

Sybase IQ assigns internal catalog proc_ids for procedures sequentially and unused proc_ids are not reused.

As procedures are dropped and created, databases created prior to Sybase IQ 12.6 may eventually reach the maximum proc_id limit of 32767, causing **CREATE PROCEDURE** to return an “Item already exists” error in Sybase IQ 12.6.

For databases created with a version prior to Sybase IQ 12.6 GA, the maximum proc_id for procedures is 32767, even if the database has been upgraded to Sybase IQ 12.6 or higher. This limit does not apply to databases created with Sybase IQ 12.6 and higher.

If the data type for the `proc_id` column is `SMALLINT`, the maximum proc_id of 32767 applies. To determine the current maximum `proc_id` value for your database, run the following query:

```
SELECT MAX (proc_id) FROM sys.sysprocedure
```

Sybase IQ 12.6 ESD7 and higher ensures that, for databases created prior to Sybase IQ 12.6, the maximum proc_id is at a level that allows **ALTER DATBASE UPGRADE** to complete. If the maximum proc_id is higher, **ALTER DATABASE UPGRADE** fails and returns the message “Database upgrade not possible”.

To resolve this issue for databases created prior to Sybase IQ 12.6, **ALTER DATABASE UPGRADE** supports a PROCEDURE ON clause in 12.6 ESD7 and higher that compacts the proc_ids by recreating all stored procedures. The syntax is **ALTER DATABASE UPGRADE PROCEDURE ON**. The PROCEDURE ON clause is ignored for databases created in 12.6 and higher.

ALTER DATABASE UPGRADE PROCEDURE ON recreates all procedures without comments. If you want the comments back in the procedures after running the command, run **ALTER PROCEDURE <procedure_name>** with your source code for the procedures that contain comments. The **sp_helptext <owner>.<procname>** command can be used to save the text of procedures with comments before running **ALTER DATABASE UPGRADE PROCEDURE ON**.

As a backup, copy the `.db` and `.log` files for the database immediately before running **ALTER DATABASE UPGRADE PROCEDURE ON**. Since only the catalog is modified during an **ALTER DATABASE UPGRADE** command, a full backup is unnecessary.

Processing Issues

Processing issues may be related to loads, queries, indexes, and table access.

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Main IQ Store Blocks Message* on page 516
- *Monitoring Disk Space Usage* on page 499
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488

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 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. The buffer caches were 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.

If you monitor paging and determine that thrashing is a problem, you can also limit the amount of thrashing during the execution of a statement which includes a query 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 `HASH_THRASHING_PERCENT` permits more paging to disk before a rollback and decreasing `HASH_THRASHING_PERCENT` permits less paging before a rollback.

Queries involving hash algorithms that executed in earlier versions of Sybase IQ may now be rolled back when the default `HASH_THRASHING_PERCENT` limit is reached. The error `Hash insert thrashing detected.` or `Hash find thrashing detected.` (SQLState QFA43, SQLCode -1001047) is reported. Take one or more of the following actions to provide the query with the resources required for execution:

- Relax the paging restriction by increasing the value of `HASH_THRASHING_PERCENT`.
- Increase the size of the temporary cache (DBA only). Keep in mind that increasing the size of the temporary cache reduces the size of the main cache.
- Attempt to identify and alleviate why Sybase IQ is misestimating 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.

To do this, 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). If you need more detailed information about the transaction that has locked the table, run the **sp_iqtransaction** stored procedure.

See also

- *How Locking Works* on page 376
- *Interactive SQL* on page 132
- *Managing Write Lock Contention on a Table* on page 506

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 are able to obtain the lock. The sample stored procedure in this section is an example of a method to manage the 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 are able to get the write lock.

The following stored procedure code manages the lock contention on a table named `dbo.event` that is used to record 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 a five second interval, and then attempts to write to the table again. The five second re-try 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;
```

See also

- *Cannot Write to a Locked Table* on page 506
- *How Locking Works* on page 376
- *Interactive SQL* on page 132

Checkpoint Hints

The default values for checkpoint time and recovery time are sufficient and do not need to be changed.

The time between checkpoints defaults to 60 minutes.

The time between checkpoints can be adjusted 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

Settings that can impact performance.

For complete information on diagnosing and resolving performance issues, see the *Performance and Tuning Guide*.

Slow Performance on a Multi-CPU or Hyperthreaded Machine

Multi-CPU or hyperthreaded machines may experience issues.

Possible Cause

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 Sybase IQ is unable to access all of the available CPUs, Sybase IQ will create too many threads and run less efficiently than it should.

Action

Start the server with **-iqnumbercpus** set to the number of CPUs available to Sybase IQ, overriding the physical number of CPUs.

For details on **start_iq** database options, see the *Utility Guide*.

Sybase Central Issues

Certain issues may be related to the operation of Sybase Central.

Some Sybase Central Fields Do Not Display

Some Sybase Central fields do not display.

Possible Cause

System is using a dark background with white text.

Action

Use the facilities your windowing system provides to change the Sybase Central display to use dark text on a white or light background.

Troubleshooting Network Communications

Communications issues may occur on Windows and with Windows-based clients.

Network software involves several different components, increasing the likelihood of problems. Although we provide some tips concerning network troubleshooting here, the primary source of assistance in network troubleshooting should be the documentation and technical support for your network communications software, as provided by your network communications software vendor.

Use diagnostic tools to obtain information you can use in diagnosing various conditions, including those described in the following sections.

See also

- *Learning Roadmap for Connections* on page 46
- *Diagnostic Tools* on page 513

Ensuring that You are Using Compatible Protocols

If you have more than one protocol stack installed on the client or server computer, you should 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*.

Ensuring that You Have Current Drivers

Old network adapter drivers are a common source of communication problems.

You should 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.

Switching off Your Computer Between Reboots

Some network adapter boards do not reset cleanly when you reboot the computer.

When you are troubleshooting, turn the computer off, wait a few seconds, and then turn it back on between reboots.

Diagnosing Your Protocol Stack Layer by Layer

If you are having problems getting your client application to communicate with a database server, you need to ensure that the client and the database server are using compatible protocol stacks.

A helpful method of isolating 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 in any way, then the data link layer is working, regardless of whether the connection is made using the same higher-layer protocols you will be using for Sybase IQ.

For example, you may want to try to connect to a disk drive on the computer running the database server from the computer running the client application.

Having verified that the data link layer is working, the next step is to verify that other applications using the same network and transport layers as Sybase IQ are working properly.

Testing a TCP/IP Protocol Stack

If you are running under TCP/IP, there are several applications that you can use to test the compatibility of the client computer and server computer TCP/IP protocol stack.

The **ping** utility provided with many TCP/IP packages is useful for testing the IP network layer.

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:

```
ping 191.72.109.12
```

at the command line prompt and wait to see if the packets are routed at all. If they are, the output will appear similar to the following:

```
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, then the computer is able to route packets to itself. This is reasonable assurance that the IP layer is set up correctly. Ask someone else running TCP/IP for their IP address and try pinging their computer.

Ensure that you can ping the computer running the database server from the client computer before proceeding.

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. The following procedure shows how 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 to see how to do this. For a typical command line Telnet program, type the following instruction 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 to see how to do this. For command line programs, you typically type the following instruction:

   ```
   telnet server_name
   ```

   where *server_name* is the name or IP address of the computer running the Telnet server process.

If a Telnet connection is established between these two machines, the protocol stack is stable and the client and server should be able to communicate using the TCP/IP link between the two computers. If a Telnet connection cannot be established, there is a problem. You should ensure that your TCP/IP protocol stack is working correctly before proceeding.

Diagnosing Wiring Problems

Faulty network wiring or connectors can cause problems that are difficult to isolate.

Try recreating problems on a similar machine with the same configuration. If a problem occurs on only one machine, the issue may be a wiring problem or a 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

If you receive the message `Unable to start — server not found` when trying to start the client, the client cannot find the database server on the network.

Check for the following problems:

- The network configuration parameters of your network driver on the client machine are different from those on the server machine. For example, two Ethernet adapter cards should be using a common frame type. For Novell NetWare, the frame type is set in the `net.cfg` file. Under Windows 98, Windows NT, and Windows 2000, the settings are accessed through 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. If this is the 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 are not installed properly or the network wiring is not installed properly.
- The network configuration parameters of your network driver are not compatible with Sybase IQ multi-user support.

"Unable to initialize any communication links" Message

If you receive the message `Unable to initialize any communication links` 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 Sybase IQ tools help you diagnose various conditions.

See also

- *Troubleshooting Network Communications* on page 509

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 in such a way that it ignores all temporary IQ file definitions in the backed up database during the restore:
   ```
   start_iq -n utility_startup_svr -c 32MB
   -x tcpip(port=1234) -iqnotemp
   ```
2. Restore the database:
   ```
   RESTORE DATABASE 'iqdemo'
   FROM '/system1/IQ15/IQ-15_3/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) 1992-2011 by Sybase, Inc.
                                     All rights reserved.
Version:                             15.3.0.5027/0490416/P/GA/MS/
                                     Windows 2000/32bit/2010-04-16
02:11:41
Time Now:                            2010-04-27 14:09:00.648
Build Time:                          2010-04-16 09:54:19
File Format:                         23 on 03/18/1999
Server mode:                         IQ 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 Temp Files:                  1
DB Blocks: 1-3200                      IQ_SYSTEM_MAIN
DB Blocks: 1045440-1055039             iq_main
Temp Blocks: 1-1600                    IQ_SYSTEM_TEMP
Create Time:                           2010-04-03 14:14:06.124
Update Time:                           2010-04-25 14:14:26.687
Main IQ Buffers:                       255, 32Mb
Temporary IQ Buffers:                  191, 24Mb
Main IQ Blocks Used:                   5915 of 11200, 52%=46Mb,
                                       Max Block#: 1051278
Temporary IQ Blocks Used:             81 of 800, 10%=0Mb, Max Block#:
161
Main Reserved Blocks Available:        1600 of 1600, 100%=12Mb
Temporary Reserved Blocks Available: 800 of 800, 100%=6Mb
IQ Dynamic Memory:                     Current: 69mb, Max: 70mb
Main IQ Buffers:                       Used: 6, Locked: 0
Temporary IQ Buffers:                  Used: 5, Locked: 0
Main IQ I/O:                      I: L899/P3 O: C3/D91/P89 D:0 C:100.0
Temporary IQ I/O:                  I: L4043/P0 O:C674/D718/P47 D:669
C:100.0
Other Txn Versions:                    0 = 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
File Encryption Status:                OFF
```

The following is a key to understanding the `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 you 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, you need to 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 will become an old version and eventually be dropped. If they rollback, the “created” data will be freed.
- `Main Reserved Blocks Available` and `Temporary Reserved Blocks Available` show the amount of reserved space that is available.
- 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.

See also

- *Main Buffer Cache Activity Message* on page 518

Interpreting Notification Messages

By default, 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. This section explains each notification message.

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 Sybase IQ inserts the number of records specified in the **NOTIFY** load option, the server sends a message like:

```
2010-05-27 13:03:49 0000000002
[20897]: 100000 Records, 2 Seconds
```

The first line shows how many rows Sybase IQ has read so far and the number of seconds taken since the last notification message to read these additional rows. Even if 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”.

See also

- *Columns in the Join Index* on page 225
- *Load and Insert Messages* on page 259
- *Message Log Wrapping* on page 11
- *Version Information in Message Logs* on page 9

Memory Message

The memory message displays information about memory usage of the Sybase IQ server.

This line in the IQ message log (`.iqmsg` file) displays memory usage information:

```
Mem: 469mb/M470
```

Table 53. Memory Usage Message

Item	Description
Mem: # mb	Current memory being used by this Sybase IQ server, in megabytes.
M# mb	The maximum number of megabytes used by this IQ server since it was started.

Main IQ Store Blocks Message

The Main IQ Store blocks message displays information about use of the blocks and buffers in the Main IQ Store.

This line in the IQ message log (`.iqmsg` file) describes the permanent Main IQ Store:

```
Main      Blks: U63137/6%, Buffers: U12578/L7
```

Table 54. Main IQ Store Blocks Message

Item	Description
U#	Number of blocks in use.
#%	Percentage of database filled.

Item	Description
Buffers: U#	Number of buffers in use. Normally this will be 100% because the buffer manager leaves buffers in memory until the buffer needs to be used 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. **Note:** This value will grow to maximum number of buffers that fit in the main buffer cache. The number increments whenever a buffer is allocated, but only decrements 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.
L#	Number of locked buffers. A locked buffer is a buffer that 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 a sort, 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 it will fail and subsequent commands may complete incorrectly. **Note:** Buffer locks do not take 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.

Recognizing 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 the section *Monitoring disk space usage*.

See also

- *Insufficient Disk Space* on page 495
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *IQ_SYSTEM_MAIN Dbspace* on page 137
- *Load Performance During Database Definition* on page 297
- *Monitoring Disk Space Usage* on page 499
- *Processing Issues* on page 504
- *Sizing Guidelines for Main and Temporary Stores* on page 139
- *Sybase IQ Stops Processing or Stops Responding* on page 488

IQ Temporary Store Blocks Message

The IQ Temporary Store blocks message displays information about use of the blocks and buffers in the Temporary IQ Store.

This line in the IQ message log (`.iqmsg` file) describes the Temporary IQ Store:

```
Temporary Blks: U273/0%, Buffers: U1987/L1960
```

Table 55. 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 will be 100% because the buffer manager leaves buffers in memory until the buffer needs to be used 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. **Note:** Objects in the temporary cache release their buffers when they are finished.
L#	Number of locked buffers. A locked buffer is a buffer that 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 a sort, 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 it will fail and subsequent commands may complete incorrectly. **Note:** Buffer locks do not take 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.

Recognizing 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 the section *Monitoring disk space usage*.

Main Buffer Cache Activity Message

The main buffer cache activity message displays information about the Main IQ Store buffer cache.

This line in the IQ message log (`.iqmsg` file) displays information about the Main IQ Store buffer cache:

```
Main      I: L331224/P22 O: D25967/P7805 C:D0
```

Table 56. Main IQ 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.

Item	Description
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 total number of bytes eligible for compression times 100. In other words, it tells how much data was 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. This 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.

See also

- *The sp_iqstatus Stored Procedure* on page 513

Temporary Buffer Cache Message

The temporary buffer cache activity message displays information about the Temporary IQ Store buffer cache.

This line in the IQ message log (`.iqmsg` file) displays information about the Temporary IQ Store buffer cache:

```
Temporary I: L25240/P8 O: D4749/P0 C:D0
```

Table 57. 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.

Item	Description
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 total number of bytes eligible for compression times 100. In other words, it tells how much data was 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. This 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 displayed 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*.

See also

- *Correlating Connection Information Between the .srvlog and .iqmsg Files* on page 528

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 does various consistency checks. However, depending on the size of your database, the check can take a long time to run.

The **sp_iqdbstatistics** stored procedure displays the database statistics collected by the most recent execution of the **sp_iqcheckdb** procedure.

See also

- *Analysis of Allocation Problems* on page 466

- *Database Connection Issues* on page 492
- *Database Verification* on page 454
- *Database Repair* on page 462
- *Index Error Repair* on page 465
- *Running out of Space During Checkpointing* on page 497
- *Server Operational Issues* on page 482
- *Starting a Server in Forced Recovery Mode* on page 471
- *Recovering Leaked Space* on page 472
- *Decision Flow for Server Recovery and Database Repair* on page 481

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 **sp_iqcheckoptions** is run, the DBA sees all options set on a permanent basis for all groups 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, run the following command:

```
sp_iqcheckoptions
```

The system table DBO.SYSOPTIONDEFAULTS contains all of the names and default values of the Sybase IQ and ASA options. You can query this table, if you need 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 tells you what statements are running on the system when you run the procedure, 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.

For details and sample output of the **sp_iqcontext** stored procedure, see *Reference: Building Blocks, Tables, and Procedures*. To match `.iqmsg` log and the **-zr** server request log entries using connection information, correlate connection information between the `.srvlog` and `.iqmsg` files.

See also

- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *Logging Server Requests* on page 522
- *Sybase IQ Stops Processing or Stops Responding* on page 488
- *Correlating Connection Information Between the .srvlog and .iqmsg Files* on page 528

Logging Server Requests

For isolating some types of problems, especially problems with queries, logging server requests is helpful.

You can enable request-level logging in two ways:

- By setting the **-zr** command-line option when you start the server.
- By calling the **sa_server_option** stored procedure, which overrides the current setting of the **-zr** command-line option.

Server requests are logged in the server log file `*.srvlog`. The **-zr** server startup option enables request-level logging of operations and sets the type of requests to log (ALL, NONE, or SQL). The **-zo** option redirects request-level logging information to a file separate from the regular log file and **-zs** limits the size of this file. See the *Utility Guide > start_iq Database Server Startup Utility*.

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.

You can enable and disable request-level logging without restarting the Sybase IQ server using the **sa_server_option** stored procedure. The following commands enable request-level logging of a limited set of requests and redirect the output to the file `sqllog.txt`:

```
call sa_server_option('request_level_logging','SQL');
call sa_server_option('request_level_log_file',
                       'sqllog.txt');
```

The following command disables request-level logging:

```
call sa_server_option('request_level_log_file','');
```

To view the current settings for the SQL log file and logging level, execute the following statement:

```
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 .

In Sybase IQ version 15.1, the request log was modified. Instead of fixed-format line prefixes, common information is 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). Request logs are much smaller now than in versions earlier than 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 the command `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
```

See also

- *Finding the Currently Executing Statement* on page 521
- *IQ Main Store and IQ Temporary Store Space Management* on page 157
- *Sybase IQ Stops Processing or Stops Responding* on page 488
- *Correlating Connection Information Between the .srvlog and .iqmsg Files* on page 528

Request Log File Analysis

The format of the output in the request log file (generated by setting the **-zr** server startup switch) changed in Sybase IQ 15.1. 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*, the **sa_server_option** system procedure in

Reference: Building Blocks, Tables, and Procedures, and *SQL Anywhere 11.0.1 > SQL Anywhere Server - SQL Usage > Monitoring and Improving Database Performance > Improving database performance > Other diagnostic tools and techniques > Request Logging*.

Connection for Collecting Diagnostic Information

The database option `DEDICATED_TASK` lets the DBA dedicate a request handling task to handling 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 Sybase Technical Support or the Sybase subsidiary in your area.

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support.

Technical Support needs information about your Sybase IQ environment in order to resolve your problem. This section describes this information, tells you how to collect it using the automated **getiqinfo** tool, and explains how to correlate information in various Sybase IQ utilities and log files.

Collecting Diagnostic Information Using getiqinfo

Sybase IQ includes a script for collecting information that Sybase Technical Support needs to diagnose problems.

The **getiqinfo** script collects information about the operating system environment, the Sybase IQ environment, and log files.

Run this script before reporting a problem to Sybase Technical Support. By doing so, you can help Sybase staff resolve your issue more quickly, with less effort on your part.

The **getiqinfo** script automatically collects information that may be needed to resolve your issue.

The **getiqinfo** script is not designed for troubleshooting Sybase IQ installations and does not provide on-site troubleshooting facilities. This script executes successfully only when the Sybase IQ environment is properly set up and the server is running.

Before You Run getiqinfo

Collect information before running the **getiqinfo** script.

Have the following information ready before running the script:

- 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 Sybase IQ message file has been renamed

If possible, leave the Sybase IQ server running, or start the server before running **getiqinfo**. This allows the script to collect internal database data that is only available when 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 Sybase IQ server. **getiqinfo** uses some IQ-specific environment variables to search for files.

The script puts 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 variable *DEST_DIR*.

Running the getiqinfo Script

On UNIX platforms, **getiqinfo** is a shell script. On Windows platforms, `getiqinfo.bat` is a batch script in the `IQ-15_3\win32` directory.

The steps vary for UNIX and Windows platforms.

1. Start the script according to your platform:
   - At the UNIX command prompt, in the `IQ-15_3/bin32` directory (on a 32-bit platform) or the `IQ-15_3/bin64` directory (on a 64-bit platform), type:
     ```
     getiqinfo.sh
     ```
   - In Windows, select Start > Run > `<install_path>\IQ-15_3\bin32\getiqinfo.bat` on a 32-bit platform or `<install_path>\IQ-15_3\bin64\getiqinfo.bat((`on a 64-bit platform.
2. As the program prompts you, 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

- Sybase IQ engine name (server name) and port number for this database server
- User ID and password with DBA privileges for this database
- The full path to the configuration file used to start the 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 Sybase Technical Support.

Information Collected by getiqinfo

The **getiqinfo** script collects all of the following information:

- Type of hardware, amount of memory, CPU type, speed, number of CPUs
- Operating system (for example, Sun Solaris 2.10)
- Swap space size
- 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-15_3/logfiles/<servername>.000n.stderr` and `IQ-15_3/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 also 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

A checklist for recording information that Technical Support may need is provided at the end of this chapter, in the unlikely event that you need to collect this information manually.

Note: 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 then rerunning 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 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. The following procedure tells you how 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.

See also

- *User Name, Connection Handle, and Connection ID* on page 520
- *Finding the Currently Executing Statement* on page 521
- *Logging Server Requests* on page 522

Another Source of Helpful Information

If you cannot resolve a problem, you may find additional help on the Sybase 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.

MySybase can be used from most Internet browsers. Point your Web browser to *Technical Documents* 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 much of this information.

Information requested	Value
Sybase IQ version (for example 15.3 GA or ESD 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), rerun command or query	

Information requested	Value
Screen snapshot of the problem, if possible	

Index

-c switch 28
-iqnotemp 513

A

Access
    ODBC configuration for 66, 67
ad hoc joins
    performance 221
Adaptive Server Enterprise
    inserting data from 52, 271
    inserting text and images 271
    unichar data type 293
    unitext data type 293
    univarchar data type 293
Advanced Security Option 339
aggregates 209
ALL permissions 320
allocation
    DBCC repair output 469
    verifying and repairing 466
allocation map
    checking allocation 456
    fixing errors 467
    inconsistencies 471
ALTER DBSPACE statement
    ADD parameter 172
    examples 172
    SIZE parameter 173
ALTER permissions 320
ALTER statement
    automatic commit 366
ALTER TABLE statement
    foreign keys 357
analytics
    Unstructured Data Analytics Option 271
analyzing output 524
ANSI code pages
    choosing 395
APPEND_LOAD option
    partitioned table 258
AppInfo connection parameter 84
archiving iqmsg message log 12
ASCII
    character set 392
    conversion on insert 283
    conversion option 277
    conversion performance 282
    data extraction 248
    data format 242
AStart connection parameter 86
AStop connection parameter 87
AUTOINCREMENT
    default 349
    negative numbers 349
    signed data types 350
    UltraLite applications 350
AutoPreCommit connection parameter 86
AutoStart connection parameter 86
AutoStop connection parameter 87
AVG function 209

B

backslashes
    Windows raw devices 153
backup log
    about 441
    location 442
BACKUP statement 412
backups
    .iqmsg file 424
    about 405
    attended 411
    concurrency 368
    concurrency issues 412
    data included in 407
    devices 409, 414
    displaying header file 438
    faster 423
    full 445
    increasing memory 447
    incremental 445
    iqmsg file 13
    Linux 410
    message log 13, 424
    message log archives 13, 424
    multiplex 406
    NULL 421
    performance issues 446
    permissions 307

privileges required 410
read-only hardware 418
recovering from errors 419
responsibilities 446
scheduling 445
specifying tape devices on NT 415
system-level 423
third party 420
unattended 411, 413
verifying 420, 429, 439
verifying incremental 439
virtual 414, 421
virtual with SAN 423
wait time 418
base tables 180
in join indexes 274
BCAST communication parameter
description 111
bcp support 242
binary
data extraction 248
binary load format
data file 263
LOAD TABLE 263
BIT data
converting 280
indexes allowed in 204
blanks
converting to NULLs 290
trimming trailing 253
BLISTENER communication parameter
description 111
BLOB data 271
BLOCK FACTOR
BACKUP statement option 417
block mode 410
block size 156
blocked write access
determining blocking writers 506
managing contention 506
blockmap 456
Broadcast communication parameter
description 111
BroadcastListener communication parameter
description 111
buffer cache
insufficient space 488
IQ UTILITIES command 489
monitor 489
buffer space
connection parameter 89
bugs
reporting 525
build number 9
bulk copy 242
bulk loading data
client data 258

C

cache
writing to 366
cache size
setting for catalog store 28
candidate keys
restrictions 344
case sensitivity
command line 22
connection parameters 83
database and server names 26
passwords 403
catalog files
growth 498
CATALOG ONLY
RESTORE option 438
catalog store
about 136
setting cache size 28
CBSize connection parameter 88
CBSpace connection parameter 89
Certificate communication parameter
description 112
Certificate_Password communication parameter
description 112
changing
collations 402
CHAR data
zero-length cells 284
character data types
Adaptive Server Enterprise unichar 293
Adaptive Server Enterprise unitext 293
Adaptive Server Enterprise univarchar 293
matching Adaptive Server Enterprise and Sybase IQ data 293
character set translation
disabling 402
error messages 397
starting a server 402
turning on and off 402

character sets
  about 391
  avoiding translation 398
  choosing 399
  connection parameter 88
  encoding 391, 392
  ISO_BINENG 392
  multibyte 396
  single-byte 392
  translation 402
  Unicode 391, 396
  UTF8 392
CharSet connection parameter 88
CHECK conditions
  columns 352
  deleting 355
  modifying 355
  tables 354
  user-defined data types 353
checklist for Technical Support 530
checkpoints
  about 382
  adjusting interval 508
  automatic and explicit 383
  in recovery 382
  in system recovery 386
  permissions 307
choosing drivers
  using the iAnywhere JDBC driver 51
  using the jConnect JDBC driver 51
client process information 84
client side
  Encryption [ENC] connection parameter 98
ClientPort communication parameter 113
clients
  direct data loading 258
CLOB data 271
CMP index 213
  recommended use 213
  restrictions 214
code pages
  about 392
  supported 392
collations
  about 391, 393, 394
  alternate 394
  ANSI 395
  changes in this version 392
  changing 402
  choosing 399
  compatibility issues 402
  CP874toUTF8 utility 391
  CREATE DATABASE statement 391
  creating a database 401, 402
  custom 391
  default 392, 401, 402
  deprecated 392
  displaying 394
  ISO_1 395
  migrating deprecated 392
  migration 391
  multibyte 396
  performance issues 402
  recommended 394
  recommended for Japanese 396
  SORT_COLLATION option 391
  sorting 393
  SORTKEY function 391
  supported 394
column delimiters
  load format option 255
column set to during load 287
column width
  insertion issues 281
columns
  adding 185
  changing 185
  defaults 344
  deleting 185
  permissions 307
  properties 348
  unrepairable errors 474
command files
  creating database objects 132
  dbisql 132
command-line switches 21
  displaying 22
  required 23
CommBufferSize connection parameter 88
CommBufferSpace connection parameter 89
committing transactions
  effect of timing on read transactions 374
CommLinks connection parameter 90
communication parameters
  about 83
  Broadcast [BCAST] 111
  BroadcastListener 111
  Certificate 112

Certificate_Password 112
ClientPort 113
DatabaseName [DBN] 114
description 122
DOBROADCAST 115
HOST 116
LDAP [LDAP] 117
LocalOnly [LOCAL] 118
LogMaxSize [LSIZE] 88, 120
LogOptions [LOPT] 120
MaxConnections 121
MYIP 122
PreFetchOnOpen 123
ReceiveBufferSize [RCVBUFSZ] 124
SendBufferSize [SNDBUFSZ] 124
ServerPort 125
SESSIONS 126
TDS 127
TIMEOUT 127, 128
communications
Encryption [ENC] connection parameter 98
parameters 109
troubleshooting 509
Compare index
See CMP index
concurrency
backups 368, 412
data definition 378
in Sybase IQ 367
insertions, deletions, and queries 377
read and write 369
configuration files
using 22
configuration parameters
overriding 487
configuring
ODBC data sources 66
connect
permission 317
ConnectFailed event handler 313
connecting
BroadcastListener [BLISTENER] communication parameter 111
character sets 398
firewalls 113
LDAP communication parameter 117
connection handle 10, 520
connection information
IQ message file 528
request log 528
srvlog file 528
connection parameters
about 83
case insensitivity 83
CharSet 88
data sources 62
default 61
Encryption [ENC] 98
IDLE 100
in connection strings 47
LANG 102
LazyClose [LCLOSE] 102
priority 84
connection profiles 52
connection strings
character sets 398
representing 48
ConnectionName connection parameter 91
connections
embedded database 59
establishing 47
examples 53
Interactive SQL 53, 54
JDBC 46
learning roadmap 46
limiting concurrent 27
local database 53, 54
logging 82
managing 311
maximum 311
overview 45
permissions for dropping 307
remote 51
restricting 428, 471
to database on foreign host 57
using data source 60
connectivity
iAnywhere JDBC driver 51
jConnect 51
consistency checking
multiplex 455
constant expression defaults 351
constraint violations
log example 260
constraints
effect on performance 342
Containment index
See WD index

conversion options
  DATE 285
  DATE format specification 285
  DATETIME 287, 288
  flat file loads 277
  performance 282
  substitution for zero-length cells 284
CONVERSION_ERROR database option 297
conversions
  between Adaptive Server Enterprise and Sybase IQ 291
  errors on import 297
  insert options 277
  on insert 276
COUNT DISTINCT
  impact on index choice 201
COUNT function 209
CREATE DATABASE statement 150
  collation 391
  IQ RESERVE parameter 172
  IQ SIZE parameter 172
  raw devices 150
  TEMPORARY RESERVE parameter 172
  TEMPORARY SIZE parameter 172
CREATE DBSPACE statement 169
  RESERVE parameter 172
  SIZE parameter 172
CREATE INDEX statement 198
CREATE JOIN INDEX statement 182, 233
CREATE permission 309
CREATE statement
  automatic commit 366
  concurrency rules 378
CREATE TABLE statement
  and command files 132
  example 179
creating
  column defaults 345
CS connection parameter 88
CS_TEXT_TYPE 271
curly braces 271
current date and time defaults 348
cursors
  connection limit 336
  hold 374, 390
  in transactions 388
  message logging 390
  ODBC configuration 67
  positioned operations 390
  scrolling 389
  sensitivity 389
custom collations 391

D

data
  client 258
  deleting 301
  duplicated 341
  exporting 241, 244
  extracting 245
  importing 241
  in transactions 372
  input and output formats 242
  invalid 341
  loading 241
data conversion error 494
data definition language
  about 131
  concurrency rules 378
data definitions
  creating 179
data extraction
  about 245
  ASCII 248
  binary 248
  binary/swap 248
  controlling access 248
  options 245
  options list 245
data integrity
  column defaults 344
  constraints 344
  overview 341
  rules in the system tables 363
data link layer
  troubleshooting 510
data manipulation language 145
data modification
  permissions 243
data source name
  ODBC 66
data sources
  about 62
  command line creation 65
  configuring 66
  connecting with 60
  ODBC 62
  UNIX 68

using with jConnect 51
data truncation error 494
data types
Adaptive Server Enterprise unichar 293
Adaptive Server Enterprise unitext 293
Adaptive Server Enterprise univarchar 293
character 293
conversion during loading 277
converting 276
converting between Adaptive Server Enterprise and Sybase IQ 291
FLOAT 293
integer 292
matching Sybase IQ and Adaptive Server Enterprise 291
money 294, 296
REAL 293
specifying in table creation 180
database
naming conflict 484
repair 453, 481
database administrator
defined 305
See also DBA 305
database server
about 3
as Windows service 21
command-line switches 21
connecting 51
name switch 24
naming at startup 25
remote 51
starting 19
starting at command prompt 20
starting from Start menu 21
starting on Windows 21
stopping 38, 42
database servers
preventing from starting 86
stopping 108
database utilities 62
DatabaseFile connection parameter 92
DatabaseName communication parameter
description 114
DatabaseName connection parameter 93
databases
block size 156
character set 397
checking consistency 426
choosing a location 152
connecting to 45, 51
creating 147
creating objects 179
creating with utility database 314
DBCC consistency checker 426
default characteristics 150
designing 131
dropping 161
embedded 83, 87
file location 149
initial size 140
initializing 147
managing with Interactive SQL 132
moving 149
moving files 431
naming 24
naming at startup 25
overview of setup 133
owner role 4
page size 154
permission to create and drop 315
permission to start 43
permissions 5, 30, 303
preallocating space 135
privileges needed to create 134
read-only restore operations 307
relative path names 151
schema creation 179
security overview 4
selective restoration 307
size 154
SQL Anywhere 136
starting 42
stopping 44
unloading 42
utility 14
validating 426
very large 15
working with objects 131
DatabaseSwitches connection parameter 94
DataSourceName connection parameter 95
DATE data type
optimizing loads 282
specifying format for conversion 285
date data types
matching Adaptive Server Enterprise and Sybase IQ data 295

DATE format
  converting two-digit dates 286
DATE index 216
  additional indexes 220
  advantages 219
  comparison to other indexes 220
  disadvantages 219
  recommended use 216
DATE option 277, 285
DATEPART
  queries 217
DATETIME
  conversion option 277
  load conversion option 287
DATETIME data type 287
  format for conversion 288
  optimizing loads 282
Datetime index
  See DTTM index
Daylight Savings Time
  message log file 14
  SQL log file 14
DBA (database administrator)
  defined 305
  role of 4
DBA authority
  about 305
  granting 306, 319
  not inheritable 325
DBASE formats 242
DBCC
  allocation verification and repair 466
  analyzing allocation problems 466
  analyzing index problems 462
  checking allocation 456
  checking indexes and allocation 455, 456
  database verification 454
  detecting allocation errors 467
  detecting index problems 475
  index verification and repair 462
  internal index checking 455
  output 458
  output messages 476
  performance 455
  repairing allocation 466, 468
  repairing indexes 462
  sample output 458
  sp_iqcheckdb interface 455
  time to run 455
DBCC_LOG_PROGRESS option 426, 458
DBF connection parameter 92
  embedded databases 59
dbisql
  command line parameters 57
  inserting data interactively 272
  logon window 58
  See also Interactive SQL 132
  specifying output format 243
  troubleshooting 494
DBISQL
  introduction 132
dbisql Java 494
DBKEY connection parameter 95
dblog utility 437
DBN communication parameter
  description 114
DBN connection parameter 93
dbo user ID
  views owned by 190
DBS connection parameter 94
dbspace
  adding on raw devices 150
  changing the size 172
  create example 172
  creating 169
  definition 135
  displaying index information 175
  displaying usage information 175
  dropping 171
  file location 149
  file location when creating 149
  IQ_SHARED_TEMP 135, 138
  IQ_SYSTEM_MSG 138
  IQ_SYSTEM_TEMP 135
  management example 172
  naming 169
  out-of-dbspace condition 157
  read-only 163
  restoring to raw device 431
  SYSTEM 135
DBSpaceLogger event 500
dbspaces
  monitoring space usage 500
  offline 467
  out of space error messages 496
  out-of-dbspace condition 495
  permissions 307

DDL
- about 131

DDL (data definition language) 7
DDL locking 378
deadlock
- detecting 489
- resolving 489

default configuration file 22
default index
- es 205

DEFAULT login policy 311, 312
DEFAULT_DISK_STRIPING option 3
defaults
- AUTOINCREMENT 349
- column 344
- connection parameters 61
- constant expressions 351
- creating 345
- creating in Sybase Central 348
- current date and time 348
- inserting 346
- loading 346
- NEWID 350
- NULL 350
- string and number 350
- USER special value 349

DELETE permissions 320
deleting
- column defaults 346

DELIMITED BY option 255
delimiters
- SELECT statement 271

Delphi
- ODBC configuration for 66

demo database 6
device types
- for databases 134

diagnostic tools 513
- checking database options 521
- checking server startup options 521
- communications issues 525
- logging server requests 522
- sa_server_option 522
- sp_iqcheckdb 520
- sp_iqcheckoptions 521
- sp_iqconnection 521
- sp_iqcontext 521
- sp_iqdbstatistics 520
- sp_iqstatus 513

Directory Services Editor
- See IQDSEDIT

DisableMultiRowFetch connection parameter 96
disconnecting 82
disk
- monitoring space usage 499
- out of space 488, 495

disk arrays
- WORM 448

disk space
- allocating 169
- indexes 203
- saving 389

DML 145
DML (Data Manipulation Language) 7
DMRF connection parameter 96
DOBROADCAST communication parameter 115
drivers
- iAnywhere JDBC driver 51
- jConnect JDBC driver 51

DROP JOIN INDEX statement 182
DROP statement
- automatic commit 366
- concurrency rules 378

DROP TABLE statement
- example 186

DROP VIEW statement
- example 191
- restriction 190

dropping
- views 190

DSN connection parameter 95
- about 62

DSS (decision support system) 2
DTTM index 216
- additional indexes 220
- advantages 219
- comparison to other indexes 220
- disadvantages 219
- recommended use 216

DumpAllThreads file 489
dumpdups
- sp_iqcheckdb option 457

dumpleaks
- sp_iqcheckdb option 457

dumpunallocs
- sp_iqcheckdb option 457

dynamic result sets 494

E

embedded databases 83, 87
  connecting 59
  Java 59
  starting 59
ENC connection parameter
  description 98
EncryptedPassword connection parameter 97
encryption
  column 339
  communications 127
  database 339
  Encryption [ENC] connection parameter 98
  FIPS 339
  RSA 339
  strong 98
Encryption connection parameter
  description 98
ENG connection parameter 96
EngineName connection parameter 96
ENP connection parameter 97
entity integrity
  about 344
  enforcing 355
environment variables
  LANG 401
  LC_ALL 401
  SQLCONNECT 62
error messages
  character set translation 397
  PIPE_NOT_CONNECTED 255
  redirecting to files 244
errors
  data conversion 297
  insertions and deletions 377
  out-of-dbspace condition 496
  transaction processing 377
  unrepairable 474
Ethernet 512
euro symbol
  1252LATIN1 collation 395, 396
event handlers
  ConnectFailed 313
events
  DBSpaceLogger 500
  monitoring disk space usage 499
  monitoring space usage 500
Excel format 242
exporting data
  about 244
  overview 241
extracting data
  about 245
  options 245
  options list 245

F

failures
  media 405
  system 405
fast projection indexes 205
fetch operation
  suppressing warnings 67
file size
  controlling 498
FileDataSourceName connection parameter 99
FileDSN
  connection parameter 62
  creating 67
  distributing 67
files
  redirecting output to 244, 245
FIPS support 339
firewalls
  BroadcastListener [BLISTENER]
    communication parameter 111
  connecting across 113
  LDAP communication parameter 117
FIXED format 242
flat files
  load conversion options 277
  loading from 254
flat FP indexes 209
FORCE_DROP option 475
forced recovery 471
  detecting duplicate blocks 457
  detecting multiply owned blocks 457
  detecting unallocated blocks 457
  procedure 473
  replacing a write server 458
  server startup failure 471
foreign keys
  existing unenforced 359
  inserting data 275
  optional 359
  restrictions 344
FoxPro format 242

FP(1) indexes 205
FP(2) indexes 205
FP(3) indexes 206
frame type 512
FROM clause
  join indexes 223
  UPDATE statement 301
functions
  BFILE 271
  for BLOB 271
  for CLOB 271
  permissions 307
  types of 8

G

getiqinfo script 525
global temporary tables
  about 181
gm switch 27
  effect on recovery 385, 454
GRANT statement
  creating groups 325
  group membership 326
  new users 317
  passwords 318
  permissions 320
  procedures 323
  WITH GRANT OPTION 322
  without password 328
GROUP BY clause
  impact on index choice 201
GROUP permissions
  not inheritable 325
groups
  creating 325
  managing 325
  membership 326
  permissions 307, 310, 327
  Sybase Central 326
  without passwords 328

H

HASH_THRASHING_PERCENT option 505
Health Insurance Portability and Accountability Act 448
HG index
  additional indexes 212
  advantages 211
  automatic creation 212
  comparison to other indexes 211
  disadvantages 211
  foreign key constraint 211
  multicolumn with NULL 212
  NULL values 212
  query performance 212
  recommended use 211
High_Group index
  See HG index
High_Non_Group index
  See HNG index
HIPAA 448
HNG index 212
  additional indexes 213
  advantages 213
  comparison to other indexes 213
  disadvantages 213
  recommended use 213
hold cursors 374, 390
HOST communication parameter 116
hyperthreading
  server switch 29

I

IANA
  port number 125
iAnywhere JDBC driver 494
  connecting to ASA databases 51
ICU library 391
IDLE connection parameter 100
importing data
  conversion errors 297
  from Adaptive Server Enterprise 270
  from pre-Version 12 IQ databases 271
  LOAD TABLE statement 254
in LOAD TABLE 287
inconsistent indexes
  repairing 465
inconsistent state 471
index types
  about 195
  criteria for choosing 200
  LF 209
  recommendations 201
  selecting 221
indexes
  about 195

adding after loading tables 222
adding and dropping 198
created automatically 182
creating 179, 198
creating TEXT indexes 220
default 205, 206
detecting logical problems 475
disk space usage 203
dropping 194
dropping corrupt 475
fast projection 205
flat FP 209
FP(1) 206
FP(1) and FP(2) 205, 206
FP(2) 206
FP(3) 206
in system tables 193
inconsistent 465
introduction 192
lookup 206
maximum unique values 505
multicolumn HG and NULL 212
parallel creation 199, 200
rebuilding 184
renaming 194
repairing 465
selecting an index type 221
sp_iqcheckdb errors 468, 475
TEXT indexes 220
too many on table 504
unrepairable errors 474
validating 194
verifying and repairing 462
insert conversion options 277
INSERT LOCATION statement 271
INSERT permissions 320
INSERT statement 222
about 268
and integrity 343
incremental 270
partitioned table 258
performance 270
VALUES option 269
inserting
column defaults 346
column width issues 281
from Adaptive Server Enterprise database 271
from older versions 277
from other databases 270
interactively 272
join index tables 274
overview 241
performance 297
primary and foreign key columns 275
See also loading data 277
selected rows 270
INT connection parameter 101
integer data types
matching Adaptive Server Enterprise and Sybase IQ 292
integrated logins
default user 80
network aspects 80
operating systems 76
permissions 308
using 78
integrity
column defaults 344
constraints 343, 344
overview 341
integrity constraint violations
log example 260
Interactive SQL
See also dbisql 132
interface libraries
connections 45
interfaces file
adding entries 52
internal build number 9
IP address 84
ping 511
IP communication parameter 116
IPv6 addresses 122
IPv6 support 339
IPX
server configuration 109
IQ PAGE SIZE 154
IQ UNIQUE
changing value of 184
performance impact 183
IQ UNIQUE constraint 352
IQ UNIQUE table option 183
IQ UTILITIES
buffer cache monitor 489
iq_bcp
deprecated 258
LOAD TABLE USING FILE replacement 258

IQ_SYSTEM_MAIN
  CREATE permission 309
  size guidelines 140
IQ_SYSTEM_MSG dbspace 138
IQ_SYSTEM_TEMP 513
  CREATE permission 309
iqdemo database 6
IQDSEDIT 52
iqdsn command 65
iqgovern switch 27
IQLANG 400
iqmc switch 26
iqmsg log 11
  archiving 12
  setting maximum size 11
  wrapping 11
iqmsg message log 8, 11
  archiving 12
  setting maximum size 11
  wrapping 11
IQMsgMaxSize server property 11
iqmsgnum server startup switch 11
IQMsgNumFiles server property 11
iqmsgsz server startup switch 11
iqnumbercpus
  server switch 29
iqpartition switch 26
iqtc switch 26
iqwmem switch 28
ISO_1 collation
  about 395
ISO_BINENG
  default collation 392, 401, 402
ISO_BINENG collation 392
isolation level
  ODBC configuration 66
isolation levels 381
ISQL
  connections 52

J

Java
  memory requirements 28
  use in Sybase IQ 46, 156
jConnect driver
  connecting to IQ databases 51
JDBC
  configuring connections 52
JDBC connectivity
  about 51
JDBC Sybase Open Client
  connections 46
join columns 230
join hierarchy 224
join indexes
  about 223
  altering columns 185
  base tables 274
  candidate keys 344
  columns in tables 225
  creating 179, 228
  creating in Sybase Central 233
  estimating benefit 239
  estimating size 239
  inserting into 274
  join hierarchy 224
  join relationships 230
  modifying underlying tables 237
  on multiplex 182
  ratio of top to related table rows 239
  synchronizing 229
join relationships
  defining 230
  specifying 232
join types 234
join virtual tables 182
joins
  linear 234
  multitable 227
  performance impact 221
  star 234
  updates using 301

K

Kerberos
  permissions 308
Kerberos authentication 339
key joins 230, 233

L

LANalyzer 512
LANG connection parameter
  description 102
LANG environment variable 401
language support
  about 391
  collations 399

multibyte character sets 396
overview 391
large object data 271
LazyClose connection parameter
description 102
LC_ALL environment variable 401
LCLOSE connection parameter
description 102
LDAP communication parameter
description 117
LDAP servers
LDAP communication parameter 117
leaked space recovery 472
LF communication parameter
description 119
LF index 209
additional indexes 210
advantages 210
comparison to other indexes 210
disadvantages 210
exceeding maximum unique values 505
recommended use 210
Links connection parameter 90
LivenessTimeout connection parameter 103
load conversions
See conversion options
load optimization 282
load options 257
load performance
lock contention 380
LOAD TABLE
BINARY 263
LOAD TABLE statement 222
about 254
BINARY FORMAT 263
binary format data file 263
integrity constraints 260
partitioned table 258
performance 282
QUOTES keyword 255
QUOTES option example 255
STRIP keyword 255
USING CLIENT FILE 258
USING FILE clause 258
loading data
Adaptive Server Enterprise data 271
ASCII conversion option 283
client data 258
column defaults 346
concurrency rules 377
conversion errors 297
conversion options 277
errors 505
file specification 254
format options 255
integrity constraint violations 259
large objects 271
logging constraint violations 259
monitoring space usage 499
named pipes 255
notification messages 515
overview 241
performance 297, 504
privileges needed 243
See also inserting 277
loading schema
recommended database size 140
loads
thread usage 299
LOB 271
LOCAL communication parameter
description 118
local temporary tables
about 181
locales
about 391, 394
character sets 397
determining 400
INSERT...LOCATION 401
setting 400
LocalOnly communication parameter
description 118
lock contention
managing 380
locking
DDL operations 378
tables 376
lockout
automatic 313
locks
managing contention 506
LOG communication parameter
description 119
log files
correlating connection information 528
server 37
LogFile
communication parameter 119

connection parameter 104
LogFormat communication parameter
description 119
logging connections 82
logical server 81
in simplex 81
logical servers 81
login attempts
exceeding limit 313
login failures 313
login management
list of procedures 312
login policies 311
option for locking 313
permissions 308
resetting 313
LOGIN_MODE database option
integrated logins 77
logins
integrated 76, 77
limiting 311
LogMaxSize communication parameter
description 88, 120
LogOptions communication parameter
description 120
lookup indexes 206
LOPT communication parameter
description 120
Lotus format 242
Low_Fast index
See LF index
LSIZE communication parameter
description 88, 120
LTO connection parameter 103

M

main store
space management 157
max_days_since_login
exceeding 313
max_failed_login_attempts
exceeding 313
MAX_TEMP_SPACE_PER_CONNECTION
option 158
MAXCONN communication parameter
description 121
MaxConnections communication parameter
description 121
MaxRequestSize communication parameter
description 122
MAXSIZE communication parameter
description 122
memory
connection limit 336
creating wired memory pool 28
for catalog store cache 28
memory message
load notification messages 516
message file
connection information 528
message log 8
backing up 13, 424
backing up archives 13, 424
Daylight Savings Time 14
IQ_SYSTEM_MSG dbspace 138
MESSAGE LOG
contents 260
example 262
message log management 8, 11
message logging 8, 11
messages
memory notification 516
out-of-dbspace condition 496
recorded in message log 8
redirecting to files 244
metadata
in catalog store 136
Microsoft Access 504
ODBC configuration for 66, 67
Microsoft Visual Basic
ODBC configuration for 66, 67
migration
collations 391
deprecated collations 392
sort keys 391
modifying
column defaults 346
modifying and deleting column defaults 346
money data types 294, 296
multibyte character sets
using 396
multiplex
consistency checks 455
creating join indexes 182
multiplex databases
backups 406
restoring 425
validating 426

multiplex servers
balancing loads 128
stopping 39
multiprocessor machines
switches 26
multithreading
during loads 299
MYIP communication parameter 122
MySybase
accessing 529
online support 529

N

named pipes 255
naming conflicts 484
national language support
about 391
collations 399
multibyte character sets 396
overview 391, 392
natural joins 233
NDIS
drivers 510
NEAREST_CENTURY option 286
net.cfg file 512
NetBIOS
server configuration 109
NetWare
network adapter settings 512
network adapters
drivers 510
network communications
command line switches 109
network number
in IPX address 115
network protocols
troubleshooting 509
NEWID
default 350
NEWPWD connection parameter 105
NOT NULL constraint 343
notification messages 515
notify count 199
NOTIFY option
INSERT statement 225
Novell client software 510
NULL 287
backups 421
conversion option 277, 290
converting to 290
default 350
inserting 269
on multicolumn HG index 212
NULL value
in multicolumn HG index 212
output 245
NULLS option
dbisql 245

O

objects
qualified names 330
ODBC
configuring data sources 66
connection parameters 83
data sources 62
initialization file for UNIX 68
UNIX support 68
ODBC connectivity
about 51
ODBC data sources
using with jConnect 51
ODBC translation driver
ODBC configuration 66
ODI drivers 510
OEM code pages
choosing 395
ON clause joins 233
Open Database Connectivity
See ODBC
operator
tasks of 446
OPERATOR authority
about 307
granting 319
option value
truncation 246, 247
options
DBCC_LOG_PROGRESS 426, 458
MAX_TEMP_SPACE_PER_CONNECTION 158
QUERY_TEMP_SPACE_LIMIT 158
setting 336
out of disk space
monitoring space usage 499
recommended actions 488, 495
output format
dbisql 243

output redirection 244, 245
owners
    about 308
    role of 4

P

page size 154
    catalog 31
    switch 31
parallel CREATE INDEX 199
params.cfg file 22
partition key 143
partitioned 241
partitioned table
    APPEND_LOAD option 258
    INSERT 258
    LOAD TABLE 258
partitioning
    definition 143
partitions
    DDL operations 144
    maximum number 144
    read-only 163
Password connection parameter 105
passwords
    assigning 308
    case sensitivity 403
    changing 318
    default 305
    expiration 311
    minimum length 318
    rules 318
    setting expiration 311
    utility database 315
    verifying 318
path names
    for databases 151
performance
    ad hoc joins 221
    effect of constraints 342
    impact of versioning 387
    indexes 192
    inserts 270
    loading data 297
    loading from flat files 278
    queries and loads 504
permissions
    command-line switches 30
    conflicts 335
    connect 317
    DBA authority 305
    dbspace management 309
    granting passwords 317
    group 325
    group membership 326
    groups 310, 327
    in Sybase Central 321
    individual 316
    inheriting 322, 325
    INSERT and DELETE, on views 334
    listing 337
    managing 303
    OPERATOR authority 307
    overview 303
    passwords 318
    PERMS ADMIN authority 307
    procedures 323
    raw devices 150
    RESOURCE authority 307
    SPACE ADMIN authority 307
    tables 309, 320
    the right to grant 322
    types of 5
    USER ADMIN authority 308
    views 191, 320
    WITH GRANT OPTION 322
PERMS ADMIN authority
    about 307
    granting 319
PFP(1) indexes 205
PFP(2) indexes 205
physical layer
    troubleshooting 512
ping
    TCP/IP 511
PIPE_NOT_CONNECTED error 255
PORT communication parameter 125
port number
    default 57
    for the database server 125
    specifying on UNIX 57
PreFetchOnOpen communication parameter
    description 123
primary keys
    AUTOINCREMENT 349
    creating 186, 187
    entity integrity 355
    inserting data 275

multicolumn 356
privileges
defining database objects 134
for inserting and deleting 243
problems
reporting 525
procedures
dynamic result sets 494
owner 308
permissions 323
permissions for creating 307
security 332
See also stored 6
system 6
product support 525
profiles
connection 52
projections
optimizing 205
PROMPT connection parameter 107
protocols
switch 32
troubleshooting 509
PWD connection parameter 105

Q

qualified object names 330
queries
concurrency rules 377
limiting concurrent 27
performance 212
performance issues 504
range predicates 218
thrashing 505
with DATEPART 217
query server
replacing a write server 458
query types
index types for 201
QUERY_TEMP_SPACE_LIMIT option 158
quotation marks
in SQL 330
QUOTES
LOAD TABLE keyword 255

R

range partitioning
definition 143
range partitions 144
raw devices 134
add dbspace 150
create database 150
naming on Windows 153
permissions 150
restoring to 431
utility database 314
RCVBUFSZ communication parameter
description 124
read-only dbspaces 162
read-only hardware
backups 418
example 448
REAL data type
matching Adaptive Server Enterprise and Sybase IQ data 293
ReceiveBufferSize communication parameter
description 124
recovery
database repair 455
database verification 455
forced 471
from system failure 490
leaked space 472
normal 454
replacing a write server 458
server 453, 481
special modes 470
system 385, 453, 454
transaction log in 386
transactions in 385, 454
versioning in 454
recycling the server 39
redirecting
output to files 244, 245
REFERENCES permissions 320
referential integrity
column defaults 344
declaring 357
enforcing 355
enforcing with existing unenforced foreign keys 359
permissions 243
RELEASE SAVEPOINT statement 383
remote data
bulk loads 258
loading 258

remote data access
  proxy tables 270
renaming database files 431
repair
  allocation 466
  database 453, 481
  indexes 475
  tables 475
request log file 524
  connection information 528
  using sa_get_request_profile 524
  using sa_get_request_times 524
request logging level 528
request-level logging 484
resetclocks
  sp_iqcheckdb option 458
resetting login policies 313
RESOURCE authority
  about 307
  granting 319
  not inheritable 325
resource planning
  iqnumbercpus switch 29
restore operations
  about 427
  displaying header file 438
  ensuring correct order 434
  excluding other users 437
  performance issues 446
  read-only selective 307
  recovering from errors 438
  SYSFILE after restore 429
  to raw device 431
  verifying backups 420, 429, 439
RESTORE statement
  about 430
  COMPATIBLE clause 439
  VERIFY clause 420, 429, 439
  verifying backups 420, 429, 439
restoring databases
  renaming files 431
  verifying backups 420, 429, 439
RetryConnectionTimeout connection parameter
    107
REVOKE statement
  about 324
rollback
  out-of-dbspace condition 157, 495
ROLLBACK statement 383
ROLLBACK TO SAVEPOINT statement 383
routers
  broadcasting over 115
ROW LOG
  contents 261
  example 262
RSA support 339

S

sa_get_request_profile
  analyzing request log file 524
sa_get_request_times
  analyzing request log file 524
SACHARSET environment variable 400
Sarbanes-Oxley Act 448
SAVEPOINT statement
  and transactions 383
savepoints
  within transactions 383
schemas
  changing 179
  creating 179
scrolling
  cursors 389
SCSI tape backups 410
security
  about 303
  Advanced Security Option 339
  column encryption 339
  database encryption 339
  Encryption [ENC] connection parameter 98
  FIPS support 339
  integrated logins 78, 79
  IPv6 support 339
  Kerberos authentication 339
  login failures 313
  procedures 323, 332
  RSA support 339
  views 332
SELECT DISTINCT projection 201
SELECT permissions 320
SELECT statement
  delimiters 271
  join indexes 223
  restrictions for view creation 190
selective restore operations 307
SendBufferSize communication parameter
  description 124

sensitivity
cursor behavior 389
server
CPU usage 488
deadlock 489
naming conflict 484
out of space 488
problems with shutdown 491
recovery 453, 481
startup failure 471
startup on Windows 486
stops processing 488
transaction log 483
unique name 484
unique port number 485
unresponsive 488, 489
See also database server
server log file 37
server objects
adding 52
server properties
IQMsgMaxSize 11
IQMsgNumFiles 11
server startup switches
-iqmsgnum 11
-iqmsgsz 11
ServerName connection parameter 96
ServerPort communication parameter 125
servers
automatic startup on boot 19
connecting 51
naming 24
recycling 39
Service Manager
starting servers 21
SESSIONS communication parameter 126
SET clause
UPDATE statement 300
shutdown
database 44
troubleshooting 491
single-byte character sets
about 392
single-node mode 182
snapshot versioning
See also versioning 365
SNDBUFSZ communication parameter
description 124
software release number 9
sort keys
migration 391
sort order
collations 391
SORT_COLLATION option 391
sorting
collations 393
SORTKEY function 391
sp_iqcheckdb
allocation mode 456
allocation verification and repair 466
analyzing allocation problems 466
analyzing index problems 462
check mode 455
checking allocation 456
checking database consistency 426
checking indexes and allocation 455, 456
database verification 454
DBCC functions 455
DBCC_LOG_PROGRESS 426
DBCC_LOG_PROGRESS option 458
dropleaks mode 458
dumpdups option 457
dumpleaks option 457
dumpunallocs option 457
index verification and repair 462
internal index checking 455
interpreting output 467
output 458
output messages 476
performance 455
repairing allocation 466, 468
repairing indexes 462
resetclocks option 458
resetting allocation maps 458
resource issues 461
sample output 458
syntax 455
time to run 455
verify mode 456
sp_iqcheckoptions stored procedure 158
sp_iqdbspace
dbspace usage information 175
sp_iqdbspaceinfo
dbspace usage information 175
sp_iqestdbspaces
estimating dbspace requirements 141
sp_iqestjoin 239
estimating join index space requirements 141

sp_iqestspace
estimating database space requirements 141
sp_iqindexinfo
displaying index information 177
sp_iqrebuildindex 465
sp_iqstatus
sample output 513
use in troubleshooting 513
sp_iqtransaction
determining blocking writers 506
SPACE ADMIN authority
about 307
granting 320
space management
IQ main store 157
IQ temporary store 157
out-of-dbspace condition 157, 495
wait-for-space condition 157, 495
specifying
drivers 51
SQL log
Daylight Savings Time 14
SQLCONNECT environment variable
connections 62
srvlog
correlating connection information 528
srvlog file
connection information 528
stack trace
generating for threads 489
location 490
star joins 234
Start parameter
embedded databases 59
START ROW ID option
INSERT statement 270
start_iq
command will not run 486
parameters 486
troubleshooting 486
start_iq utility 20
starting 44
StartLine connection parameter 108
startup
allocation error 471
checkpoint error 471
resolving a failure 471
troubleshooting hints 483
startup parameters 21
startup script 20
startup utility 20
stored procedures
about 6
granting permissions to execute 334
Sybase IQ 6
string and number defaults 350
STRIP
LOAD TABLE keyword 255
strong encryption
Encryption [ENC] connection parameter 98
subtransactions
and savepoints 383
SUM function 209
Sybase Central 44
adding users to groups 327
altering tables 185
and permissions 321
column constraints 353
column defaults 348
creating dbspaces 171
creating groups 326
creating tables 179
creating users 317
display problems 502, 509
dropping views 191
introduction 132
starting 53
Sybase Control Center
adding members to groups 327
creating dbspaces 171
creating groups 326
creating users 317
Sybase IQ
matching data types with Adaptive Server Enterprise 291
Sybase IQ 12.5 servers 494
synchronizing
about 229
multiplex 182
SYS group 329
SYSCOLAUTH view
permissions 338
SYSCOLUMN table
integrity 363
SYSDUMMY table
permissions 338
SYSFILE table
file_name after restore 429

SYSFOREIGNKEY table
    integrity 363
SYSGROUP table
    permissions 338
SYSGROUPS view
    permissions 338
SYSPROCAUTH view
    permissions 338
SYSPROCPERM table
    permissions 338
SYSTABAUTH view
    permissions 338
SYSTABLE table
    integrity 363
SYSTABLEPERM table
    permissions 338
SYSTEM dbspace 135
system failure
    recovering from 490
system tables 7
    about 188
    indexes in 193
    permissions 337
    showing 159
    SYSFILE 429
    users and groups 337
system unresponsive 488
system views
    integrity 363
    permissions 337
system-level backups 423
SYSUSERAUTH view
    permissions 338
SYSUSERLIST view
    permissions 338
SYSUSERPERM table
    permissions 338
SYSUSERPERMS view
    permissions 338
SYSVIEW view
    view information 192

T

table partition 143
table-level versioning
    about 368
    See also versioning 368
tables
    adding keys to 186, 187
    altering 185
    blocked access 506
    corrupt 475
    creating 179
    dropping 186
    group owners 328
    join relationships 230
    joining multiple 227
    loading 254
    locking 376
    managing blocked access 506
    moving to new dbspace 309
    owner 308
    permissions 307
    qualified names 328, 330
    read-only 163
    See also information in system tables 7
    unrepairable errors 474
tablespaces
    allocation 3
tape devices
    for backup 415
TCP/IP
    BroadcastListener [BLISTENER]
        communication parameter 111
    connecting across firewalls 113
    LDAP communication parameter 117
    server configuration 109
    server port number 125
    testing 511
    troubleshooting 510
TDS communication parameter 127
Technical Support
    checklist 530
    MySybase 529
    online help 529
    reporting problems to 525
Telnet
    TCP/IP testing 511
TEMP_EXTRACT_NULL_AS_EMPTY option
    250
TEMP_EXTRACT_NULL_AS_ZERO option 250
temporary dbfiles 513
temporary storage
    option to save space 389
temporary store
    space management 157
temporary tables
    about 181

loading 181
versioning 375
TEXT indexes 220
creating 220
thrashing
HASH_THRASHING_PERCENT option 505
threads
generating a stack trace 489
not enough 501
use during loads 299
TIME data type
optimizing loads 282
time data types
matching Adaptive Server Enterprise and Sybase IQ data 295
TIME index 216
additional indexes 220
advantages 219
comparison to other indexes 220
disadvantages 219
recommended use 216
TIMEOUT communication parameter 127
TO communication parameter 127
top table 223
size and performance 227
trace
generating for threads 489
trailing blanks
trimming 253
transaction log
in system recovery 386
renaming 437
transaction processing
about 365
transactions
about 365
cursors in 388
definition 365
ending 366
in recovery 385, 454
rolling back 385
savepoints 383
starting 366
subtransactions and savepoints 383
translation driver
ODBC configuration 66
trimming trailing blanks 253
troubleshooting 481
common problems 512
database connection 492
dbisql 494
processing issues 504
protocols 509
resource issues 495
server operation 482
Sybase Central 509
wiring problems 512

U

UID connection parameter 109
UNC connection parameter 108
Unconditional connection parameter 108
unenforced foreign keys 359
unichar Adaptive Server Enterprise data type 293
Unicode
character sets 391, 396
Unilib library 391
UNIQUE constraints 351
unitext Adaptive Server Enterprise data type 293
univarchar Adaptive Server Enterprise data type 293
UNIX
ODBC support 68
Unstructured Data Analytics Option 271
UPDATE statement
using 300
using join operations 301
USER ADMIN authority
about 308
user authentication
permissions 308
USER authority
granting 320
user IDs
creating 317
default 305
deleting 324
in message log 10
listing 337
managing 303
user names
See user IDs
USER special value
default 349
user-defined data types
CHECK conditions 353

Userid connection parameter 109
users
  adding to groups 327
  creating in Sybase Central 317
  creating individual 316
  locking 311
  locking out 313
  login failures 313
  permissions 308
  unlocking 313
using column defaults 344
util_db.ini file 315
utilities
  CP874toUTF8 391
  transaction log 437
utility database
  about 14
  connecting 316
  password to create databases 315
  security 314
  setting password 315
  starting 314
utility programs
  iq_bcp deprecated 258

V

VALUES option
  INSERT statement 269
VARCHAR data
  zero-length cells 284
variable-length data transfer 410
VERIFY communication parameter 128
verifying backups 420, 429, 439
  error reporting 440
  incremental 439
  progress reporting 440
verifying passwords 318
version string 9
versioning
  about 365, 368
  at table level 368
  cursors and 389
  in recovery 454
  in system recovery 386
  isolation levels 381
  performance impact 387
  temporary tables 375
views
  creating 189
  deleting 190, 191
  differences from permanent tables 188
  inserting and deleting 190
  modifying 190
  owner 308
  permissions 191, 307, 309
  security 332
  SELECT statement restrictions 190
  using 190
  working with 188
Virtual Backup
  decoupled 422
  encapsulated 421
Visual Basic
  ODBC configuration for 66, 67
VLDB
  managing 15

W

WarehouseArchitect
  about 131
WD index 214
  advantages 216
  disadvantages 216
  recommended use 214
WHERE clause
  impact on index choice 201
  join indexes 223
  UPDATE statement 301
Windows Service 19
wired memory
  setting iqwmem switch 28
wiring
  troubleshooting 512
WITH GRANT OPTION clause 322
WORD index 214
working with column defaults in Sybase Central 348
WORM storage 448
write server
  replacing 458

X

XML format 242

Y

year 2000
  conversion options 286

Z

zeros
  converting to NULL 290
zr log file 524