

**SYBASE®**

System Administration Guide

**Sybase® IQ**

12.7

DOCUMENT ID: DC00170-01-1270-01

LAST REVISED: June 2006

Copyright © 1991-2006 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase, SYBASE (logo), ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Advantage Database Server, Afaria, Answers Anywhere, Applied Meta, Applied Metacomputing, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, ASE/P, Avaki, Avaki (Arrow Design), Avaki Data Grid, AvantGo, Backup Server, BayCam, Beyond Connected, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DataWindows .NET, DB-Library, dbQueue, Dejima, Dejima Direct, Developers' Workbench, DirectConnect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, EII Plus, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, ExtendedAssist, ExtendedSystems, ExtendedView, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, irLite, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Legion, Logical Memory Manager, M2M Anywhere, Mach Desktop, Mail Anywhere, Studio, Mainframe Connect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Anywhere, M-Business Channel, M-Business Network, M-Business Suite, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, mFolio, Mirror Activator, ML Query, MobiCATS, MobileQ, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASiS, OASiS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniQ, OmniSQL Access Module, OmniSQL Toolkit, OneBridge, Open Biz, Open Business Interchange, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, Pharma Anywhere, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power++, Power Through Knowledge, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Pylon, Pylon Anywhere, Pylon Application Server, Pylon Conduit, Pylon PIM Server, Pylon Pro, QAnywhere, Rapport, Relational Beans, RemoteWare, RepConnector, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RFID Anywhere, RW-DisplayLib, RW-Library, SAFE, SAFE/PRO, Sales Anywhere, Search Anywhere, SDF, Search Anywhere, Secure SQL Server, Secure SQL Toolkit, Security Guardian, ShareSpool, ShareLink, SKILS, smart.partners, smart.parts, smart.script, SOA Anywhere Trademark, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, Stage III Engineering, Startup Com, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase IQ, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Unimill, Unisep, Unistring, URK Runtime Kit for UniCode, Vodafone, Viewer, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, XP Server, XTNDAccess and XTNDConnect are trademarks of Sybase, Inc. or its subsidiaries. 05/06

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

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

# Contents

About This Book .....	xxiii	
<b>CHAPTER 1</b>	<b>Overview of Sybase IQ System Administration .....</b>	<b>1</b>
	Introduction to Sybase IQ.....	2
	System administration tasks.....	3
	Security overview .....	4
	Types of users .....	4
	Granting permissions .....	5
	Tools for system administration.....	5
	The database server .....	6
	Catalogs and IQ .....	6
	The IQ Store .....	7
	The IQ Temporary Store .....	7
	The Catalog Store .....	7
	Adaptive Server Anywhere and Sybase IQ .....	7
	Concurrent operations.....	8
	Multiplex capability .....	8
	Stored procedures.....	10
	Sybase IQ stored procedures.....	11
	Adaptive Server Enterprise system and catalog procedures ..	13
	Catalog stored procedures .....	15
	System tables and views.....	16
	Commands and functions .....	21
	Types of SQL statements .....	22
	Functions.....	22
	Message logging .....	23
	Version information in the iqmsg file .....	23
	Collation information in the iqmsg file.....	24
	Connection information in the iqmsg file .....	24
	Controlling the size of the iqmsg file.....	25
	Moving the iqmsg file in Sybase Central .....	25
	Insert notifications in the iqmsg file .....	25
	The utility database .....	26
	Managing very large databases .....	26

Managing memory use.....	27
Managing data loads .....	27
Managing processing threads .....	28
Managing disk space.....	28
Intermediate versioning .....	28
Creating databases .....	29
Creating indexes .....	29
Optimizing queries.....	29
Schema design.....	29
UNION ALL views .....	30
<b>CHAPTER 2</b>	
<b>Running Sybase IQ.....</b>	<b>31</b>
Starting the database server .....	31
Starting servers with the startup utility.....	33
Rules for running multiplex servers .....	34
Starting servers with Sybase Central .....	35
Starting servers from the Windows Start menu.....	44
Restarting servers when the Windows host is restarted .....	44
Running the server as a Windows service .....	45
Using command-line switches.....	45
Naming the server and databases .....	47
Controlling performance and memory from the command line	49
Controlling permissions from the command line .....	53
Setting a maximum Catalog page size.....	55
Setting up a client/server environment.....	55
Starting a server in forced recovery mode .....	57
Starting a server from DBISQL.....	58
Starting multiple servers or clients on the same machine .....	58
Monitoring server activity .....	59
Stopping the database server .....	62
When to stop and restart the server.....	62
How to stop the server .....	62
Who can stop the server .....	65
Shutting down operating system sessions .....	66
Restarting multiplex servers with Sybase Central.....	66
Starting and stopping databases.....	66
Starting the asiqdemo database .....	68
Starting and stopping Sybase Central.....	68
Connecting a plug-in .....	69
Stopping Sybase Central.....	70
<b>CHAPTER 3</b>	
<b>Sybase IQ Connections.....</b>	<b>71</b>
Introduction to connections .....	72

How connections are established.....	72
How connection parameters work.....	74
Connection parameters are passed as connection strings .....	75
Saving connection parameters in ODBC data sources .....	75
Connecting from Sybase Central or Interactive SQL .....	76
Opening the Connect dialog .....	77
Specifying a driver for your connection .....	78
Working with the Connect dialog.....	79
Connection shortcuts in Sybase Central .....	80
Creating server objects .....	80
Creating connection profiles .....	83
Simple connection examples .....	85
Connecting to the sample database from Sybase Central or DBISQL .....	86
Connecting to a database on your own machine from Sybase Central or Sybase IQ .....	87
Connecting to other databases from DBISQL.....	88
Connecting using command line utilities .....	89
Connecting to an embedded database .....	91
Connecting using a data source.....	93
Using default connection parameters.....	94
Connecting from Sybase IQ utilities .....	95
Working with ODBC data sources.....	95
Where data sources are held .....	96
Creating and editing ODBC data sources .....	97
Configuring ODBC data sources .....	99
Using file data sources.....	104
Using ODBC data sources on UNIX .....	105
Connecting to a database using OLE DB .....	106
OLE DB providers .....	107
Connecting from ADO .....	108
Connection parameters .....	109
Connection parameter tips .....	109
How Sybase IQ makes connections .....	110
Steps in establishing a connection .....	110
Locating the interface library .....	111
Assembling a list of connection parameters.....	112
Locating a server.....	113
Locating the database .....	115
Server name caching for faster connections .....	116
Interactive SQL connections .....	117
Connecting from other databases .....	117
Testing that a server can be found.....	118
Using an integrated login .....	118

Using integrated logins.....	119
Security concerns: unrestricted database access .....	123
Setting temporary public options for added security .....	124
Network aspects of integrated logins .....	125
Creating a default integrated login user .....	125
Login procedures and IQ User Administration .....	126
Disconnecting and dropping connections .....	126
Connection logging .....	127
Troubleshooting startup, shutdown, and connections.....	127
What to do if you can't start Sybase IQ .....	127
What to do if you can't connect to a database.....	129
Stopping a database server in an emergency (UNIX).....	131
Using the UNIX kill command to stop DBISQL.....	131
Resolving problems with your DBISQL window on UNIX.....	132
<b>CHAPTER 4</b>	
<b>Connection and Communication Parameters.....</b>	<b>133</b>
Connection parameters.....	133
AppInfo connection parameter [App].....	134
AutoPreCommit connection parameter [AutoPreCommit].....	136
AutoStart connection parameter [Astart] .....	137
AutoStop connection parameter [Astop].....	137
CharSet connection parameter [CS] .....	138
CommBufferSize connection parameter [CBSIZE] .....	138
CommLinks connection parameter [Links] .....	139
ConnectionName connection parameter [CON].....	141
DatabaseFile connection parameter [DBF] .....	141
DatabaseName connection parameter [DBN] .....	143
DatabaseSwitches connection parameter [DBS] .....	144
DataSourceName connection parameter [DSN] .....	144
DBKEY connection parameter [DBKEY] .....	145
DisableMultiRowFetch connection parameter [DMRF] .....	145
EngineName connection parameter [ENG] .....	146
EncryptedPassword connection parameter [ENP] .....	146
Encryption connection parameter [ENC] .....	147
FileDataSourceName connection parameter [FileDSN].....	149
Idle connection parameter [IDLE].....	149
Integrated connection parameter [INT].....	150
Language connection parameter [LANG].....	151
LazyClose connection parameter [LCLOSE].....	151
LivenessTimeout connection parameter [LTO] .....	152
LogFile connection parameter [LOG] .....	153
Password connection parameter [PWD] .....	153
PrefetchBuffer connection parameter [PBUF] .....	154
PrefetchRows connection parameter [PROWS].....	155

ServerName connection parameter [ENG].....	155
StartLine connection parameter [START] .....	156
Unconditional connection parameter [UNC] .....	156
Userid connection parameter [UID] .....	156
Network communications parameters.....	157
Broadcast communication parameter [BCAST].....	158
BroadcastListener communication parameter [BLISTENER] .....	159
Certificate communication parameter.....	159
Certificate_Password communication parameter .....	160
ClientPort parameter [CPort] .....	160
DatabaseName communication parameter [DBN] .....	161
DoBroadcast parameter [DBROAD].....	161
DLL parameter .....	162
Host parameter (IP).....	163
LDAP communication parameter [LDAP] .....	164
LocalOnly communication parameter [LOCAL] .....	164
LogFile communication parameter [LOG] .....	165
LogFormat communication parameter [LF] .....	165
LogMaxSize communication parameter [LSIZE].....	166
LogOptions communication parameter [LOPT] .....	167
MaxConnections communication parameter [MAXCONN]....	168
MaxRequestSize communication parameter [MAXSIZE]....	168
MyIP parameter [ME] .....	168
PreFetchOnOpen communication parameter.....	169
ReceiveBufferSize communication parameter [RCVBUFSZ] .....	170
SendBufferSize communication parameter [SNDBUFSZ] ....	170
ServerPort parameter (PORT) .....	170
Sessions parameter .....	171
TDS parameter .....	172
Timeout parameter [TO] .....	172
VerifyServerName parameter [Verify] .....	173
 <b>CHAPTER 5</b>	
<b>Working with Database Objects .....</b>	<b>175</b>
Building your Sybase IQ databases .....	176
Designing your database.....	176
Tools for working with database objects .....	176
A step-by-step overview of database setup .....	178
Extending data definition privileges.....	179
Selecting a device type .....	180
Allocating space for databases .....	180
Working with database objects .....	183
Creating a database with SQL .....	183
Reserving space to handle out-of-space conditions.....	193
Creating databases with multiplex functionality.....	193

Multiplex management without Sybase Central .....	201
Mixed-version multiplexes .....	203
Synchronizing query servers .....	204
Persistent objects .....	206
Managing multiplex servers .....	207
Deleting query servers .....	211
Updating multiplex databases .....	212
Resolving static collisions.....	212
Setting database options .....	213
Showing system objects in a database .....	214
Disconnecting from a database .....	215
Dropping a database .....	216
Working with dbspaces .....	216
Adding dbspaces.....	216
Dropping a dbspace .....	224
Dropping dbspaces from a multiplex database .....	225
Dbspace management example .....	228
Working with tables .....	239
Creating tables .....	239
Altering tables.....	245
Dropping tables .....	246
Creating primary and foreign keys .....	247
Table information in the system tables.....	249
Working with views.....	249
Creating views.....	250
Using views .....	251
Modifying views .....	252
Permissions on views.....	252
Deleting views .....	253
Views in the system tables .....	253
Working with indexes .....	254
Introduction to indexes .....	254
Creating indexes .....	255
Indexes in the system tables .....	255
Validating indexes .....	256
Renaming indexes.....	256
Removing indexes.....	256
<b>CHAPTER 6</b>	
<b>Using Sybase IQ Indexes .....</b>	<b>257</b>
Overview of indexes .....	257
Sybase IQ index types .....	258
Benefits over traditional indexes .....	261
Creating Sybase IQ indexes .....	261
The CREATE INDEX statement.....	262

Creating an index with Sybase Central .....	263
Creating indexes concurrently .....	263
Choosing an index type.....	264
Number of unique values in the index .....	265
Types of queries .....	265
Indexing criteria: disk space usage .....	268
Data types in the index.....	269
Combining index types .....	270
Sybase IQ index types .....	270
Default column index.....	270
The Low_Fast (LF) index type.....	271
The High_Group (HG) index type.....	273
The High_Non_Group (HNG) index type .....	275
The Compare (CMP) index type.....	276
The Containment (WD) index type .....	277
The Date (DATE), Time (TIME), and Datetime (DTTM) index types	
279	
Optimizing performance for ad hoc joins.....	285
Selecting an index .....	285
Adding column indexes after inserting data .....	286
Using join indexes .....	287
Join indexes improve query performance .....	287
Loading considerations for join indexes .....	287
How join indexes are used for queries .....	288
Relationships in join indexes .....	288
When a join becomes ad hoc .....	288
Join hierarchy overview.....	288
Columns in the join index .....	289
The join hierarchy in query resolution .....	290
Multiple table joins and performance.....	292
Steps in creating a join index .....	293
Synchronizing join indexes .....	295
Defining join relationships between tables .....	296
Issuing the CREATE JOIN INDEX statement .....	299
Creating a join index in Sybase Central .....	301
Types of join hierarchies .....	301
Modifying tables included in a join index .....	304
Inserting or deleting from tables in a join index .....	305
Table versioning controls access to join indexes .....	306
Estimating the size and benefit of a join index .....	307
Using sp_iqestjoin to estimate join index size .....	307
Weighing join index benefit by comparing numbers of rows .	308

Import and export overview.....	309
Import and export methods .....	310
Input and output data formats .....	311
Permissions for modifying data .....	312
Scheduling database updates .....	313
Exporting data from a database .....	313
Using output redirection .....	313
Data extraction options.....	315
Bulk loading data using the LOAD TABLE statement.....	324
Interpreting notification messages.....	343
Logging integrity constraint violations .....	347
Using the INSERT statement .....	351
Inserting specified values row by row .....	352
Inserting selected rows from the database .....	353
Inserting from a different database.....	354
Importing data interactively .....	357
Inserting into tables of a join index.....	357
Inserting into primary and foreign key columns.....	358
Partial-width insertions .....	359
Partial-width insertion rules .....	360
Converting data on insertion .....	364
Inserting data from pre-Version 12 Sybase IQ .....	366
Load conversion options .....	366
Column width issues .....	370
Optimizing date and time loads .....	371
Using the ASCII conversion option.....	373
The DATE option.....	374
The DATETIME conversion option.....	377
Working with NULLS .....	380
Other factors affecting the display of data.....	381
Matching Adaptive Server Enterprise data types .....	382
Unsupported Adaptive Server Enterprise data types .....	382
Adaptive Server Enterprise data type equivalents .....	383
Handling conversion errors on data import .....	387
Tuning bulk loading of data .....	387
Improving load performance during database definition .....	387
Setting server startup options.....	388
Adjusting your environment at load time .....	388
Reducing Main IQ Store space use in incremental loads.....	390
Understanding thread use during loads .....	391
Changing data using UPDATE.....	391
Deleting data .....	393
<b>CHAPTER 8      Using Procedures and Batches.....</b>	<b>395</b>

Overview of procedures .....	396
Benefits of procedures .....	396
Introduction to procedures .....	397
Creating procedures .....	398
Altering procedures .....	399
Calling procedures .....	400
Copying procedures in Sybase Central.....	401
Deleting procedures .....	401
Permissions to execute procedures .....	402
Returning procedure results in parameters .....	402
Returning procedure results in result sets.....	403
Introduction to user-defined functions .....	404
Creating user-defined functions .....	404
Calling user-defined functions .....	405
Dropping user-defined functions .....	406
Permissions to execute user-defined functions.....	406
Introduction to batches.....	407
Control statements .....	408
Using compound statements.....	409
Declarations in compound statements .....	409
Atomic compound statements .....	410
Structure of procedures.....	411
SQL statements allowed in procedures.....	411
Declaring parameters for procedures.....	412
Passing parameters to procedures .....	413
Passing parameters to functions .....	413
Returning results from procedures.....	414
Returning a value using the RETURN statement.....	414
Returning results as procedure parameters .....	415
Returning result sets from procedures .....	417
Returning multiple result sets from procedures.....	418
Returning variable result sets from procedures.....	419
Using cursors in procedures .....	420
Cursor management overview .....	420
Cursor positioning .....	421
Using cursors on SELECT statements in procedures .....	422
Errors and warnings in procedures .....	424
Default error handling in procedures .....	425
Error handling with ON EXCEPTION RESUME .....	427
Default handling of warnings in procedures .....	429
Using exception handlers in procedures .....	430
Nested compound statements and exception handlers .....	432
Using the EXECUTE IMMEDIATE statement in procedures .....	434
Transactions and savepoints in procedures.....	434

Tips for writing procedures .....	434
Check if you need to change the command delimiter .....	435
Remember to delimit statements within your procedure .....	435
Use fully-qualified names for tables in procedures.....	435
Specifying dates and times in procedures.....	436
Verifying procedure input arguments are passed correctly ...	436
Hiding the contents of procedures, functions, and views .....	436
Statements allowed in batches .....	437
Using SELECT statements in batches .....	438
Using IQ UTILITIES to create your own stored procedures.....	438
How IQ uses the IQ UTILITIES command .....	439
Requirements for using IQ UTILITIES.....	439
Choosing procedures to call.....	441
Numbers used by IQ UTILITIES.....	441
Testing your procedures.....	441
<b>CHAPTER 9</b>	
<b>Ensuring Data Integrity .....</b>	<b>443</b>
Data integrity overview .....	443
How data can become invalid .....	444
Ensuring valid data.....	444
Changing database contents.....	445
Data integrity tools.....	446
SQL statements for implementing integrity constraints .....	447
Using column defaults.....	447
Creating column defaults.....	448
Modifying and deleting column defaults .....	449
Working with column default values .....	449
Working with column defaults in Sybase Central .....	451
Date, time, and timestamp defaults.....	452
USER defaults.....	453
The IDENTITY or AUTOINCREMENT default .....	453
The NULL default .....	454
String and number defaults .....	454
Constant expression defaults .....	455
USING table and column constraints .....	455
Using UNIQUE constraints on columns or tables.....	456
Using IQ UNIQUE constraint on columns .....	456
Using CHECK conditions on columns .....	456
Defining CHECK conditions on user-defined data types.....	458
Working with column constraints in Sybase Central .....	458
Using CHECK conditions on tables.....	459
Modifying and deleting CHECK conditions.....	459
Declaring entity and referential integrity .....	460
Declaring entity integrity .....	460

CHAPTER 10	Enforcing entity integrity .....	461
	If a client application breaches entity integrity.....	461
	Declaring referential integrity.....	462
	Losing referential integrity .....	466
	Controlling concurrent operations .....	466
	Disabling referential integrity checking .....	469
	Integrity rules in the system tables.....	469
<b>CHAPTER 10</b>	<b>Transactions and Versioning.....</b>	<b>471</b>
	Overview of transactions and versioning .....	471
	Introduction to transactions .....	472
	Introduction to concurrency .....	474
	Multiplex table version logging .....	476
	Communications between servers .....	476
	Introduction to versioning .....	477
	Versioning prevents inconsistencies .....	485
	How locking works .....	485
	Locks for DML operations .....	486
	Locks for DDL operations .....	486
	Primary keys and locking .....	488
	Managing locks .....	489
	Displaying active locks .....	489
	Managing lock contention.....	489
	Isolation levels.....	490
	Checkpoints, savepoints, and transaction rollback .....	492
	Checkpoints.....	492
	Savepoints within transactions .....	493
	Rolling back transactions .....	495
	System recovery.....	496
	How transaction information aids recovery .....	497
	Performance implications.....	498
	Overlapping versions and deletions .....	499
	Cursors in transactions .....	500
	Cursors and versioning .....	500
	Cursor sensitivity .....	501
	Cursor scrolling .....	501
	Hold cursors .....	501
	Positioned operations.....	502
	Cursor command syntax and examples .....	502
	Controlling message logging for cursors .....	502
<b>CHAPTER 11</b>	<b>International Languages and Character Sets.....</b>	<b>503</b>
	Introduction to international languages and character sets.....	503

Sybase IQ international features .....	503
Using the default collation .....	504
Character set questions and answers .....	505
Understanding character sets in software.....	505
Pieces in the character set puzzle.....	506
Language issues in client/server computing .....	506
Code pages in Windows.....	507
Multibyte character sets .....	513
Sorting characters using collations.....	514
International aspects of case sensitivity .....	515
Understanding locales.....	516
Introduction to locales .....	516
Understanding the locale language.....	517
Understanding the locale character set.....	518
Understanding collations .....	521
Choosing collations .....	521
Displaying collations.....	522
Supplied and recommended collations .....	523
Alternate collations .....	525
ANSI or OEM?.....	526
Notes on ANSI collations.....	527
Notes on OEM collations.....	528
Using multibyte collations.....	530
Understanding character set translation .....	531
Character translation for database messages.....	531
Connection strings and character sets .....	532
Avoiding character-set translation .....	533
Collation internals.....	534
Comment lines .....	535
The title line .....	535
The collation section.....	536
The Encodings section .....	538
The Properties section .....	538
International language and character set tasks .....	539
Finding the default collation.....	539
Configuring your character set environment .....	539
Determining locale information.....	540
Setting locales .....	541
Creating a database with a named collation .....	543
Starting a database server using character set translation ...	544
Creating a custom collation .....	544
Creating a database with a custom collation.....	546
Changing a database from one collation to another.....	547
Compatibility issues .....	547

	Performance issues .....	548
<b>CHAPTER 12</b>	<b>Managing User IDs and Permissions.....</b>	<b>549</b>
	An overview of database permissions.....	549
	DBA authority overview .....	550
	RESOURCE authority overview .....	552
	Ownership permissions overview .....	552
	Table and views permissions overview .....	552
	Group permissions overview .....	553
	Multiplex permissions overview.....	553
	Server command-line permission options .....	556
	Managing IQ user accounts and connections .....	556
	Multiplex login management .....	561
	Collisions in login management.....	563
	Utility database server security .....	564
	Defining the utility database password.....	564
	Permission to execute file administration statements .....	565
	Managing individual user IDs and permissions.....	566
	Creating new users .....	567
	Changing a password.....	568
	Granting DBA and resource authority .....	569
	Granting permissions on tables and views .....	570
	Granting users the right to grant permissions .....	572
	Granting permissions on procedures .....	572
	Revoking user permissions .....	573
	Managing groups .....	574
	Creating groups.....	575
	Granting group membership to users .....	576
	Permissions of groups.....	576
	Referring to tables owned by groups.....	577
	Groups without passwords .....	577
	Special groups.....	578
	Database object names and prefixes.....	579
	Using views and procedures for extra security .....	581
	Using views for tailored security.....	581
	Using procedures for tailored security .....	583
	Using procedures to disable connections.....	583
	How user permissions are assessed .....	584
	Managing the resources connections use.....	584
	Database options that govern user resources.....	584
	Other settings that affect user resources .....	586
	Users and permissions in the system tables .....	586

<b>CHAPTER 13</b>	<b>Transport-Layer Security .....</b>	<b>589</b>
	About transport-layer security .....	590
	FIPS 140-2 certification .....	590
	About public-key cryptography .....	591
	Client architecture .....	592
	Digital certificates .....	592
	The role of digital certificates.....	593
	Using chains of certificates.....	594
	Server authentication .....	595
	Setting up transport-layer security .....	595
	Invoking transport-layer security .....	596
	Self-signed certificates .....	597
	Certificate authorities .....	600
	Certificate chains.....	601
	Enterprise root certificates .....	602
	Creating the certificates.....	603
	Using the signed certificates .....	604
	Globally signed certificates .....	605
	Getting server-authentication certificates.....	606
	Using a global certificate as a server certificate .....	607
	Verifying certificate fields .....	608
	Verifying fields in certificate chains .....	608
	Using a globally-signed certificate as an enterprise certificate	608
	Using transport-layer security for web services .....	610
<b>CHAPTER 14</b>	<b>Data Backup, Recovery, and Archiving.....</b>	<b>613</b>
	Backup protects your data .....	613
	Backing up your database.....	614
	Types of data stores.....	614
	Types of backups .....	615
	Selecting archive devices.....	619
	Preparing for backup .....	620
	Concurrency and backups.....	622
	The BACKUP statement.....	623
	Backup examples .....	629
	Recovery from errors during backup .....	630
	After you complete a backup.....	631
	Performing backups with non-Sybase products .....	631
	Virtual Backups .....	632
	Types of Virtual Backups.....	632
	Virtual Backup with SAN snapshot or shadow hardware .....	634
	Performing system-level backups .....	634
	Shutting down the database (non-multiplex) .....	635
	Shutting down the database (multiplex) .....	635

Backing up the right files .....	635
Restoring from a system-level backup .....	637
Validating your database.....	637
Restoring your databases .....	639
Before you restore .....	639
The RESTORE statement .....	642
Restoring in the correct order .....	647
Reconnecting after you restore .....	649
Renaming the transaction log after you restore .....	649
Validating the database after you restore.....	650
Restore requires exclusive write access .....	650
Displaying header information.....	651
Recovery from errors during restore .....	651
Backups and symbolic links (UNIX only).....	651
Unattended backup .....	652
Getting information about backups and restores .....	653
Locating the backup log .....	653
Content of the backup log .....	654
Maintaining the backup log.....	655
Recording dbspace names.....	655
Determining your data backup and recovery strategy.....	656
Scheduling routine backups .....	657
Designating backup and restore responsibilities .....	658
Improving performance for backup and restore .....	658
Backing up and restoring the multiplex .....	660
Multiplex server migration and failover.....	663
Strategy for loss of a machine.....	664
Loss of the write server's machine .....	666
Migration.....	666
Recovery from loss of a multiplex server .....	669
Write server failure where some disk resources are lost.....	669
Archiving data with read-only hardware .....	670
Dbspace access modes .....	670
Effect of altering dbspaces on access modes .....	671
Transitions between access modes .....	672
Using read-only hardware .....	674
<b>CHAPTER 15</b>	
<b>Sybase IQ as a Data Server.....</b>	<b>677</b>
Client/server interfaces to Sybase IQ.....	677
Configuring IQ Servers with DSEDITION .....	679
Sybase applications and Sybase IQ.....	685
Open Client applications and Sybase IQ.....	685
Setting up Sybase IQ as an Open Server .....	685
System requirements .....	685

	Starting the database server as an Open Server .....	686
	Configuring your database for use with Open Client.....	687
	Characteristics of Open Client and jConnect connections .....	687
	Servers with multiple databases.....	690
<b>CHAPTER 16</b>	<b>Accessing Remote Data.....</b>	<b>693</b>
	Sybase IQ and remote data .....	694
	Requirements for accessing remote data.....	694
	Working with remote servers.....	696
	Working with external logins.....	703
	Working with proxy tables .....	704
	Example: a join between two remote tables.....	708
	Accessing multiple local databases.....	709
	Sending native statements to remote servers .....	710
	Using remote procedure calls (RPCs).....	711
	Multiplex servers and remote data access.....	712
	Transaction management and remote data .....	713
	Remote transaction management overview .....	713
	Restrictions on transaction management .....	714
	Internal operations .....	714
	Query parsing.....	715
	Query normalization .....	715
	Query preprocessing .....	715
	Server capabilities .....	715
	Complete passthrough of the statement .....	716
	Partial passthrough of the statement.....	717
	Troubleshooting remote data access .....	718
	Features not supported for remote data.....	718
	Performance limitations for Java Virtual Machine .....	719
	Case sensitivity .....	719
	Connectivity problems .....	720
	General problems with queries.....	720
	Queries blocked on themselves .....	720
	Managing remote data access connections .....	721
<b>CHAPTER 17</b>	<b>Server Classes for Remote Data Access.....</b>	<b>723</b>
	Server classes overview .....	723
	JDBC-based server classes .....	724
	Configuration notes for JDBC classes.....	724
	Server class asajdbc .....	724
	Server class asejdbc .....	726
	ODBC-based server classes .....	727
	Defining ODBC external servers .....	727

<b>CHAPTER 18</b>	<b>Automating Tasks Using Schedules and Events.....</b>	<b>739</b>
	Introduction to scheduling and event handling.....	740
	Understanding schedules.....	741
	Defining schedules .....	742
	Understanding events .....	742
	Choosing a system event .....	743
	Defining trigger conditions for events .....	744
	Understanding event handlers .....	745
	Developing event handlers.....	746
	Schedule and event internals.....	747
	How the database server checks for events .....	747
	How the database server checks for scheduled times .....	748
	How event handlers are executed.....	748
	Scheduling and event handling tasks.....	749
	Adding a schedule or event to a database .....	749
	Adding a manually-triggered event to a database .....	749
	Triggering an event handler .....	750
	Debugging an event handler .....	750
	Retrieving information about an event or schedule .....	751
<b>APPENDIX A</b>	<b>XML in the Database.....</b>	<b>753</b>
	Introduction to XML .....	753
	Source code and Javadoc.....	754
	Installing XML in Sybase IQ .....	754
	References .....	754
	An overview of XML .....	754
	Using XML in the Sybase IQ database .....	760
	Mapping and storage.....	760
	Client or server considerations.....	761
	Accessing XML in SQL.....	761
	XML parsers .....	762
	Storing XML data in a Sybase IQ database .....	763
	The OrderXml class for Order documents.....	763
	Creating and populating SQL tables for Order data .....	766
	Using the element storage technique .....	767
	A customizable example for different result sets.....	770

	The ResultSet document type .....	771
	The ResultSetXml class for result set documents .....	775
	Using the element storage technique.....	778
	XML ResultSet documents: invalid XML characters .....	782
APPENDIX B	<b>Data Access Using JDBC .....</b>	<b>787</b>
	JDBC overview.....	787
	Choosing a JDBC driver.....	788
	JDBC program structure.....	789
	Server-side JDBC features.....	790
	Differences between client- and server-side JDBC connections..	792
	Establishing JDBC connections .....	792
	Connecting from a JDBC client application using jConnect ..	793
	Establishing a connection from a server-side JDBC class ....	797
	Notes on JDBC connections .....	799
	Using JDBC to access data .....	800
	Preparing for the examples .....	801
	Inserts, updates, and deletes using JDBC .....	801
	Passing arguments to Java methods .....	803
	Queries using JDBC.....	804
	Using prepared statements for more efficient access .....	806
	Inserting and retrieving objects .....	807
	Miscellaneous JDBC notes .....	808
	Using the Sybase jConnect JDBC driver .....	808
	Versions of jConnect supplied with Sybase IQ.....	809
	The jConnect driver files.....	809
	Installing jConnect system objects into a database.....	810
	Loading the driver.....	810
	Supplying a URL for the server .....	811
	Creating distributed applications .....	813
	Implementing the Serializable interface .....	814
	Importing the class on the client side .....	815
	A sample distributed application.....	815
	Other features of distributed applications.....	816
APPENDIX C	<b>Debugging Logic in the Database.....</b>	<b>817</b>
	Introduction to debugging in the database .....	817
	Debugger features.....	817
	Requirements for using the debugger .....	818
	Tutorial 1: Getting started with the debugger .....	819
	Lesson 1: Connect to a database and start the debugger ....	819
	Tutorial 2: Debugging a stored procedure.....	820

Display stored procedure source code in the debugger.....	820
Set a breakpoint .....	821
Run the procedure.....	821
Inspect and modify variables.....	822
Tutorial 3: Debugging a Java class .....	823
Prepare the database .....	823
Display Java source code into the debugger .....	823
Set a breakpoint .....	824
Run the method.....	825
Step through source code .....	825
Inspect and modify variables.....	826
Common debugger tasks .....	827
Writing debugger scripts .....	829
sybase.asa.procdebug.DebugScript class .....	829
sybase.asa.procdebug.IDebugAPI interface .....	830
sybase.asa.procdebug.IDebugWindow interface .....	834
<b>Index .....</b>	<b>837</b>



# About This Book

<b>Subject</b>	Sybase® IQ is a high-performance decision support server designed specifically for data warehouses and data marts. This book, <i>Sybase IQ System Administration Guide</i> , presents the concepts and procedures necessary for the administration of Sybase IQ.
<b>Audience</b>	This guide is for system and database administrators and for anyone who needs to set up or manage Sybase IQ. Familiarity with relational database systems and introductory user-level experience with Sybase IQ is assumed. Use this guide with other manuals in the documentation set.
<b>How to use this book</b>	The following table shows which chapters fit a particular interest or need.

**Table 1: Guide to using this book**

To learn how to...	Read this chapter...
Understand the role of an Sybase IQ administrator	Chapter 1, “Overview of Sybase IQ System Administration”
Start and stop an IQ database server, and set up user connections	Chapter 2, “Running Sybase IQ”
Set up user connections	Chapter 3, “Sybase IQ Connections” and Chapter 4, “Connection and Communication Parameters”
Create an Sybase IQ database	Chapter 5, “Working with Database Objects”
Select Sybase IQ indexes	Chapter 6, “Using Sybase IQ Indexes”
Load data into your database	Chapter 7, “Moving Data In and Out of Databases”
Create procedures and batches	Chapter 8, “Using Procedures and Batches”
Specify constraints on the data in your tables	Chapter 9, “Ensuring Data Integrity”
Understand how transactions work	Chapter 10, “Transactions and Versioning”
Set up your database for the language you work in	Chapter 11, “International Languages and Character Sets”
Add users and assign them privileges	Chapter 12, “Managing User IDs and Permissions”
Protect the confidentiality and integrity of data passing between a client and the Sybase IQ server	Chapter 13, “Transport-Layer Security”
Back up and restore databases	Chapter 14, “Data Backup, Recovery, and Archiving”
Set up IQ for use as an Open Server™	Chapter 15, “Sybase IQ as a Data Server”
Configure Sybase IQ to access remote data	Chapter 16, “Accessing Remote Data” Chapter 17, “Server Classes for Remote Data Access”

---

<b>To learn how to...</b>	<b>Read this chapter...</b>
Automate database administration tasks using scheduling and event handling	Chapter 18, “Automating Tasks Using Schedules and Events”
Use Java tools to access XML documents in Sybase IQ	Appendix A, “XML in the Database”
Use JDBC to access data	Appendix B, “Data Access Using JDBC”
Use the Sybase debugger	Appendix C, “Debugging Logic in the Database”

## Related documents

Documentation for Sybase IQ:

- *Introduction to Sybase IQ*

Read and try the hands-on exercises if you are unfamiliar with Sybase IQ or the Sybase Central™ database management tool.

- *New Features in Sybase IQ 12.7*

Read just before or after purchasing Sybase IQ for a list of new features.

- *Sybase IQ Performance and Tuning Guide*

Read to understand query optimization, design, and tuning issues for very large databases.

- *Sybase IQ Reference Manual*

Read for a full description of the SQL language, stored procedures, data types, and system tables supported by Sybase IQ.

- *Sybase IQ Troubleshooting and Recovery Guide*

Read to solve problems and perform system recovery and database repair.

- *Sybase IQ Error Messages*

Refer to IQ error messages (referenced by SQLCode, SQLState, and Sybase error code) and SQL preprocessor errors and warnings.

- *Sybase IQ Utility Guide*

Read for Sybase IQ utility program reference material, such as available syntax, parameters, and options.

- *Sybase IQ Installation and Configuration Guide*

Read the edition for your platform before and while installing Sybase IQ, when migrating to a new version of Sybase IQ, or when configuring Sybase IQ for a particular platform.

- *Sybase IQ Release Bulletin*

Read just before or after purchasing Sybase IQ for last minute changes to the product and documentation. Read for help if you encounter a problem.

- *Large Objects Management in Sybase IQ*

Read to understand storage and retrieval of Binary Large Objects (BLOBs) and Character Large Objects (CLOBs) within the Sybase IQ data repository. You need a separate license to install this product option.

- *Encrypted Columns in Sybase IQ*

Read to understand the use of user encrypted columns within the Sybase IQ data repository. You need a separate license to install this product option.

---

**Sybase IQ and Adaptive Server Anywhere**

Because Sybase IQ is an extension of Adaptive Server® Anywhere, a component of SQL Anywhere® Studio, IQ supports many of the same features as Adaptive Server Anywhere. The IQ documentation set refers you to SQL Anywhere Studio documentation where appropriate.

---

Documentation for Adaptive Server Anywhere:

- *Adaptive Server Anywhere Programming Guide*

Intended for application developers writing programs that directly access the ODBC, Embedded SQL™, or Open Client™ interfaces, this book describes how to develop applications for Adaptive Server Anywhere.

- *Adaptive Server Anywhere Database Administration Guide*

Intended for all users, this book covers material related to running, managing, and configuring databases and database servers.

- *Adaptive Server Anywhere SQL Reference Manual*

Intended for all users, this book provides a complete reference for the SQL language used by Adaptive Server Anywhere. It also describes the Adaptive Server Anywhere system tables and procedures.

You can also refer to the Adaptive Server Anywhere documentation in the SQL Anywhere Studio 9.0.2 collection on the Sybase Product Manuals Web site. To access this site, go to Product Manuals at <http://www.sybase.com/support/manuals/>.

---

<b>Other sources of information</b>	Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:
	<ul style="list-style-type: none"> <li>• The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.</li> <li>• The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.</li> </ul>
	<p>Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.</p>
	<p>Refer to the <i>SyBooks Installation Guide</i> on the Getting Started CD, or the <i>README.txt</i> file on the SyBooks CD for instructions on installing and starting SyBooks.</p>
<b>Sybase certifications on the Web</b>	<ul style="list-style-type: none"> <li>• The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.</li> </ul>
	<p>To access the Sybase Product Manuals Web site, go to Product Manuals at <a href="http://www.sybase.com/support/manuals/">http://www.sybase.com/support/manuals/</a>.</p>
	<ul style="list-style-type: none"> <li>• Infocenter is an online version of SyBooks that you can view using a standard Web browser. To access the Infocenter Web site, go to Sybooks Online Help at <a href="http://infocenter.sybase.com/help/index.jsp">http://infocenter.sybase.com/help/index.jsp</a>.</li> </ul>
<b>Sybase certifications on the Web</b>	<p>Technical documentation at the Sybase Web site is updated frequently.</p>
	<p>❖ <b>Finding the latest information on product certifications</b></p> <ol style="list-style-type: none"> <li>1 Point your Web browser to Technical Documents at <a href="http://www.sybase.com/support/techdocs/">http://www.sybase.com/support/techdocs/</a>.</li> <li>2 Select Products from the navigation bar on the left.</li> <li>3 Select a product name from the product list and click Go.</li> <li>4 Select the Certification Report filter, specify a time frame, and click Go.</li> <li>5 Click a Certification Report title to display the report.</li> </ol>

- ❖ **Finding the latest information on component certifications**
  - 1 Point your Web browser to Availability and Certification Reports at <http://certification.sybase.com/>.
  - 2 Either select the product family and product under Search by Product; or select the platform and product under Search by Platform.
  - 3 Select Search to display the availability and certification report for the selection.
- ❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

  - 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
  - 2 Click MySybase and create a MySybase profile.

**Sybase EBFs and software maintenance**

- ❖ **Finding the latest information on EBFs and software maintenance**
  - 1 Point your Web browser to the Sybase Support Page at <http://www.sybase.com/support>.
  - 2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
  - 3 Select a product.
  - 4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.
  - 5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

**Syntax conventions**

This documentation uses the following syntax conventions in syntax descriptions:

- 
- **Keywords** SQL keywords are shown in UPPER CASE. However, SQL keywords are case insensitive, so you can enter keywords in any case you wish; SELECT is the same as Select which is the same as select.
  - **Placeholders** Items that must be replaced with appropriate identifiers or expressions are shown in *italics*.
  - **Continuation** Lines beginning with ... are a continuation of the statements from the previous line.
  - **Repeating items** Lists of repeating items are shown with an element of the list followed by an ellipsis (three dots). One or more list elements are allowed. If more than one is specified, they must be separated by commas.
  - **Optional portions** Optional portions of a statement are enclosed by square brackets. For example:

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

It indicates that the *savepoint-name* is optional. The square brackets should not be typed.

- **Options** When none or only one of a list of items must be chosen, the items are separated by vertical bars and the list enclosed in square brackets. For example:

```
[ ASC | DESC ]
```

It indicates that you can choose one of ASC, DESC, or neither. The square brackets should not be typed.

- **Alternatives** When precisely one of the options must be chosen, the alternatives are enclosed in curly braces. For example:

```
QUOTES { ON | OFF }
```

It indicates that exactly one of ON or OFF must be provided. The braces should not be typed.

**Table 2: Typographic conventions**

Item	Description
Code	SQL and program code is displayed in a mono-spaced (fixed-width) font.
User entry	Text entered by the user is shown in bold serif type.
<i>emphasis</i>	Emphasized words are shown in italic.
<i>file names</i>	File names are shown in italic.
database objects	Names of database objects, such as tables and procedures, are shown in bold, san-serif type in print, and in italic online.

**The sample database**

Sybase IQ includes a sample database used by many of the examples in the IQ documentation.

The sample database represents a small company. It contains internal information about the company (employees, departments, and financial data), as well as product information (products), sales information (sales orders, customers, and contacts), and financial information (fin\_code, fin\_data).

The sample database is held in a file named *asiqdemo.db*, located in the directory *\$ASDIR/demo* on UNIX systems and *%ASDIR%\demo* on Windows systems.

**Accessibility features**

This document is available in an HTML version that is specialized for accessibility. You can navigate the HTML with an adaptive technology such as a screen reader, or view it with a screen enlarger.

Sybase IQ 12.7 and the HTML documentation have been tested for compliance with U.S. government Section 508 Accessibility requirements. Documents that comply with Section 508 generally also meet non-U.S. accessibility guidelines, such as the World Wide Web Consortium (W3C) guidelines for Web sites.

For information about accessibility support in the Sybase IQ plug-in for Sybase Central, see “Using accessibility features” in *Introduction to Sybase IQ*. The online help for this product, which you can navigate using a screen reader, also describes accessibility features, including Sybase Central keyboard shortcuts.

**Configuring your accessibility tool**

You might need to configure your accessibility tool for optimal use. Some screen readers pronounce text based on its case; for example, they pronounce ALL UPPERCASE TEXT as initials, and MixedCase Text as words. You might find it helpful to configure your tool to announce syntax conventions. Consult the documentation for your tool and see “Using screen readers” in *Introduction to Sybase IQ*.

---

For information about how Sybase supports accessibility, see Sybase Accessibility at <http://www.sybase.com/accessibility>. The Sybase Accessibility site includes links to information on Section 508 and W3C standards.

For a Section 508 compliance statement for Sybase IQ, go to Sybase Accessibility at <http://www.sybase.com/products/accessibility>.

**If you need help**

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

# Overview of Sybase IQ System Administration

## About this chapter

This chapter provides a brief introduction to Sybase IQ and an overview of IQ system administration.

## Contents

Topic	Page
Introduction to Sybase IQ	2
System administration tasks	3
Security overview	4
Tools for system administration	5
The database server	6
Catalogs and IQ	6
Stored procedures	10
System tables and views	16
Commands and functions	21
Message logging	23
The utility database	26
Managing very large databases	26

## Introduction to Sybase IQ

Sybase IQ is a high-performance decision support server designed specifically for data warehousing. This cross-platform product runs on several popular Unix platforms, as well as Windows XP and Windows 2000.

Sybase IQ is part of the Adaptive Server family that includes **Adaptive Server Enterprise** for enterprise transaction and mixed workload environments and **Adaptive Server 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 Adaptive Server 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, so that you can 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. They are typically 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** capability is designed to manage large query loads across multiple nodes. Multiplex capability supports many users, each executing complex queries against the shared database. With each server running Sybase IQ and a shared disk subsystem connected to all servers, you can match I/O operations to CPU capabilities and scale the user community past the limits of a single server. Sybase IQ multiplexes also offer higher availability, as failures of a single node leave all other nodes up and running. You can attach as many servers as the disk subsystem supports. For more about multiplex capability, see “Multiplex capability” on page 8.

**Learning more about  
Sybase IQ**

This book explains how you manage a Sybase IQ system. It is intended for database administrators and others who need to understand database creation, load, and operation issues. “About This Book” describes other documentation helpful to administrators.

## System administration tasks

Typically, the database administrator (DBA) is responsible for the tasks listed on the left side of the following table. Look at the right side of the table to see where these tasks are explained in this or other manuals.

**Table 1-1: Administrative tasks**

<i>If you want to know how to...</i>	<i>Look in...</i>
Install and configure Sybase IQ for your platform	<i>Sybase IQ Installation and Configuration Guide</i>
Start and stop the database server	Chapter 2, “Running Sybase IQ”
Set up or manage user connections	Chapter 3, “Sybase IQ Connections” and Chapter 4, “Connection and Communication Parameters”
Create a Sybase IQ database (with or without multiplex capability)	Chapter 5, “Working with Database Objects”
Determine appropriate indexes for your users' queries	Chapter 6, “Using Sybase IQ Indexes”
Load data into your database	Chapter 7, “Moving Data In and Out of Databases”
Create procedures and batches	Chapter 8, “Using Procedures and Batches”
Ensure the integrity of data in your tables	Chapter 9, “Ensuring Data Integrity”
Understand how transactions impact concurrency	Chapter 10, “Transactions and Versioning”
Set up your database for the language you work in	Chapter 11, “International Languages and Character Sets”
Add users, assign them privileges, and manage logins	Chapter 12, “Managing User IDs and Permissions”
Back up and restore databases	Chapter 14, “Data Backup, Recovery, and Archiving”
Tune Sybase IQ for maximum performance	<i>Sybase IQ Performance and Tuning Guide</i>

Monitor IQ performance	<i>Sybase IQ Performance and Tuning Guide</i>
Set up IQ for use as an Open Server	Chapter 15, “Sybase IQ as a Data Server”
Configure Sybase IQ to access remote data	Chapter 16, “Accessing Remote Data” Chapter 17, “Server Classes for Remote Data Access”
Automate database administration tasks using scheduling and event handling	Chapter 18, “Automating Tasks Using Schedules and Events”
Use Java tools to access XML documents in Sybase IQ	Appendix A, “XML in the Database”
Use Java in a Sybase IQ database	Appendix B, “Data Access Using JDBC” Appendix C, “Debugging Logic in the Database”
Diagnose IQ server conditions	Chapter 1, “Troubleshooting Hints” in <i>Sybase IQ Troubleshooting and Error Messages Guide</i>
Verify or recover your database	Chapter 2, “System Recovery and Database Repair” in <i>Sybase IQ Troubleshooting and Error Messages Guide</i>

## Security overview

The DBA is responsible for maintaining database security. Sybase IQ provides security controls by means of the privileges you can assign to users.

## Types of users

Sybase IQ recognizes three categories of users for each IQ database:

- 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.

## Granting permissions

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.
- Alter the structure of database objects.
- Insert, update, or delete data.
- Select (view) data.
- Execute procedures.

The DBA can grant any type of authority to any user. Sometimes other users can grant authority as well. For more information on what users can do, and how the DBA manages users, see Chapter 12, “Managing User IDs and Permissions.”

## Tools for system administration

To help you manage your database, Sybase IQ provides two primary tools:

- **Sybase Central** is an application for managing Sybase databases. It helps you manage database objects and 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.
- **DBISQL**, also called Interactive SQL, is an application that allows you to enter SQL statements interactively and send them to a database. DBISQL has a window-like user interface on all platforms. Sybase IQ supports both a Java-based DBISQL, and the C-based DBISQL provided in previous releases.

The *Introduction to Sybase IQ* explains how to use Sybase Central and DBISQL 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.

In addition to these tools, Sybase IQ provides a number of stored procedures that perform system management functions. See “Stored procedures” on page 10 for more information. 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 Chapter 18, “Automating Tasks Using Schedules and Events” for details.

In addition to the stored procedures, Sybase IQ provides a tool to monitor the performance of its buffer caches. This monitor collects statistics on the buffer cache, memory, and I/O functions taking place within Sybase IQ, and stores them in a log file. See *Sybase IQ Performance and Tuning Guide* for details.

A few administrative tasks, such as selecting a collation, rely on command-line utilities. These utilities are discussed in other chapters of this book, and described in the *Sybase IQ Reference Manual*.

## The database server

The **database server** is the “brain” of your Sybase IQ system. Users access data through the database server, never directly. Requests for information from a database are sent to the database server, which carries out the instructions.

## Catalogs and IQ

An IQ database is a joint data store consisting of at least three parts:

- The permanent IQ Store
- The IQ Temporary Store
- The Catalog Store

When you create an IQ database, all three stores are created automatically. You create IQ databases using the procedures described in Chapter 5, “Working with Database Objects.”

An IQ database may also have an IQ Message Store and (on a multiplex query server) an IQ Local Store.

## The IQ Store

The **IQ Store** is the set of Sybase IQ tables. You can have one or more permanent IQ Stores, each in a separate database. Each IQ Store includes a set of tables that organize your data. The table data is stored in indexes, which are structured to allow rapid response to various types of analytical queries on very large quantities of data. The IQ Store is sometimes referred to as the IQ Main Store.

## The IQ Temporary Store

The **IQ Temporary Store** consists of a set of temporary tables. The database server uses them for sorting and other temporary processing purposes.

## The Catalog Store

The **Catalog Store** contains all of the information required to manage an IQ database. This information, which includes system tables and stored procedures, resides in a set of tables that are compatible with Adaptive Server Anywhere. These tables contain the **metadata** for the IQ database. Metadata describes the layout of the IQ tables, columns, and indexes. The Catalog Store is sometimes referred to simply as the Catalog.

## Adaptive Server Anywhere and Sybase IQ

The Catalog Store closely resembles an Adaptive Server Anywhere store. Adaptive Server Anywhere is a relational database system that can exist with or without IQ. You may have Adaptive Server Anywhere-style tables in your Catalog Store along with your IQ tables, or you may have a separate Adaptive Server Anywhere database.

Anywhere tables have a different format than IQ tables. While the commands you use to create objects in an Anywhere database are 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 the Sybase IQ Reference Manual for operations in the IQ Store.*

This book explains how you manage your IQ Store and its associated Catalog Store. If you have an Anywhere database, or if you have Anywhere-style tables in your Catalog Store, see the Adaptive Server Anywhere documentation for details of how to create, maintain, and use them.

For a summary of differences between Sybase IQ and Adaptive Server Anywhere, see Appendix A, “Compatibility with Other Sybase Databases,” in the *Sybase IQ Reference Manual*.

## 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.

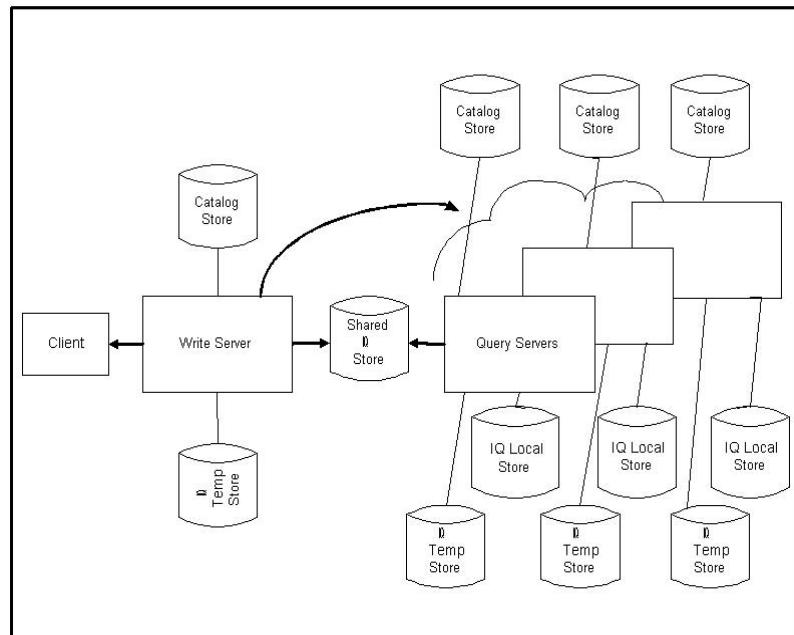
Sybase IQ keeps your database consistent during these concurrent operations by maintaining multiple versions of table data. To understand this approach, see Chapter 10, “Transactions and Versioning.”

## Multiplex capability

One way to manage a large query load would be to copy the entire IQ database to multiple machines. Extra servers could each run the same database, but after changes at one of the servers, many of the files would have to be copied to the other servers again. Once the IQ Store became very large, you would have to purchase lots of extra disk capacity, and copying the database would take a long time.

IQ multiplex capability provides an effective alternative, where the IQ Store cost is reduced by sharing it, taking advantage of a shared disk subsystem. Sybase IQ's vertical storage, compression and bitwise technology reduce I/O requirements dramatically, making it possible to have many systems sharing the disk array(s) before contention degrades performance. You gain additional CPU power and memory space for processing queries by attaching more systems to the shared disk array. Multiplex capability lets a user perform loads while other users query the database.

**Figure 1-1: IQ multiplex architecture**



Each set of an IQ Temporary Store and Catalog Store comprise one **server**.

In an IQ multiplex, one server can load or update the database while the others submit queries. The read-only servers submitting queries are called **query servers**. The updating server is known as the **write server**. By allowing only a single write server, Sybase IQ multiplex eliminates the overhead and scalability limits of a distributed lock manager.

Each server of a multiplex may start, stop, and operate independent of all the others. Failure of any server does not prevent the other servers from operating. There is no need to stop a write server. It is only necessary to stop query servers in extraordinary circumstances:

- Configuring a new write server

- Restoring the query server local store from backup
- Performing forced recovery/drop leaks on a local store on the query server
- Starting the write server in single-node mode

The servers share a common IQ Store. Users may also create an IQ Local Store on each query server, in which they can create and store persistent objects. Such objects are visible to all users of that query server, but not to users on other servers of the multiplex. For details see “Populating persistent tables in multiplex databases” on page 241.

The following sections provide more details about multiplex operation:

- “Rules for running multiplex servers” on page 34
- “Running the Sybase IQ Agent” on page 35
- “Starting all servers in a multiplex” on page 41
- “Creating databases with multiplex functionality” on page 193
- “Synchronizing query servers” on page 204
- “Updating multiplex databases” on page 212
- “Creating tables in multiplex databases” on page 239
- “Multiplex table version logging” on page 476
- “Backing up and restoring the multiplex” on page 660
- “Multiplex server migration and failover” on page 663

## Stored procedures

Sybase IQ **stored procedures** help you manage your system. Stored procedures give you information about your database and users, and carry out various operations on the database. This section briefly describes the stored procedures. For more information, see the *Sybase IQ Reference Manual*.

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 `asiqdemo` database, it adds a user to `asiqdemo`.

You can also create your own stored procedures. See Chapter 8, “Using Procedures and Batches” for details.

---

**Note** Statements shown in examples generally use the asiqdemo database, a sample database installed as part of Sybase IQ. For a diagram of this database’s structure, see *Introduction to Sybase IQ*.

---

## Sybase IQ stored procedures

The following procedures work specifically on the IQ Store. They are owned by the DBA user ID.

---

**Note** Stored procedures that produce size information assume that the database was created with the default block size, as described in “Block size” in *Sybase IQ Performance and Tuning Guide*. If a database was created with a non-default block size, the output from the following stored procedures is inaccurate: sp\_iqestjoin, sp\_iqestdbspaces, sp\_iqestspace.

---

**Table 1-2: Stored Procedures for the IQ Store**

Procedure name	Purpose
sp_iqaddlogin	Adds a new Sybase IQ user account, and specifies the number of current logins the user may have and the password expiration interval
sp_iqcheckdb	Checks the validity of the current database and repairs indexes
sp_iqcolumn	Displays columns in a database and information about them
sp_iqconnection	Shows information about connections and versions, including which users are using temporary dbspace, which users are keeping versions alive, what the connections are doing inside Sybase IQ, connection status, and database version status
sp_iqconstraint	Lists referential integrity constraints for the specified table or column
sp_iqcontext	Displays the currently executing statement for each active connection, or for a single connection
sp_iqcursorinfo	Displays detailed information about cursors currently open on the server
sp_iqdatatype	Displays information about system data types and user-defined data types
sp_iqdbsize	Gives the size of the current database
sp_iqdbspace	Lists detailed information including usage, properties, and types of data on each dbspace.

<b>Procedure name</b>	<b>Purpose</b>
sp_iqdbspaceinfo	Displays the number of blocks user per index per main or local dbspace for one or all dbspaces.
sp_iqdbstatistics	Reports results of the most recent sp_iqcheckdb
sp_iqdroplogin	Drops an IQ login
sp_iqestjoin	Estimates the space needed to create join indexes for the tables you specify
sp_iqestdbspaces	Estimates the number and size of dbspaces needed for a given total index size
sp_iqestspace	Estimates the amount of space needed to create a database, based on the number of rows in the underlying database tables
sp_iqeevent	Displays information about system and user-defined events
sp_iqhelp	Displays information about system and user-defined objects and data types
sp_iqindex	Lists indexes and information about them. Omitting the parameter lists all indexes in the database. Specifying the <i>table_name</i> parameter lists indexes for this table only.
sp_iqindexinfo	Displays the number of blocks user per index per main or local dbspace for a given object.
sp_iqindex_alt	Same as sp_iqindex, except that for multicolumn indexes it lists one row for each column.
sp_iqindexsize	Gives the size of the specified index
sp_iqjoinindex	Displays information about join indexes
sp_iqjoinindexsize	Gives the size of the specified join index
sp_iqlistexpiredpasswords	Lists user accounts with expired passwords
sp_iqlistlockedusers	Lists locked user accounts
sp_iqlistpasswordexpirations	Lists user accounts, their password creation dates, and number of days password is valid from creation date
sp_iqlocklogin	Locks an IQ user account so that the user cannot log in
sp_iqlocks	Shows information about locks in the IQ Store and the Catalog Store
sp_iqmodifyadmin	Enables or disables Sybase IQ User Administration, and sets defaults for user account characteristics in the IQ_SYSTEM_LOGIN_INFO_TABLE system table
sp_iqmodifylogin	Modifies the maximum number of connections or the password expiration interval for a single user or all users
sp_iqpassword	Adds or changes a password for an IQ login account
sp_iqpkkeys	Displays information about primary keys and primary key constraints by table, column, table owner, or for all IQ tables in the database
sp_iqprocedure	Displays information about system and user-defined procedures
sp_iqprocparm	Displays information about stored procedure parameters, including result set variables and SQLSTATE/SQLCODE error values

<b>Procedure name</b>	<b>Purpose</b>
sp_iq_process_login	When the LOGIN_PROCEDURE database option is set to sp_iq_process_login, determines which users can log on
sp_iqrelocate	Relocates specified tables and indexes on main dbspaces in relocate mode to main or local dbspaces in readwrite mode.
sp_iqrename	Renames user-created tables, views, columns, indexes, constraints (unique, primary key, foreign key, and check), stored procedures, and functions
sp_iqshowpsexec	Displays information about the settings of database options that control the priority of tasks and resource usage for connections
sp_iqspaceinfo	Lists the number of blocks used per object per dbspace.
sp_iqspaceused	Shows information about space available and space used in IQ Store and IQ Temporary Store
sp_iqstatus	Displays a variety of IQ status information about the database
sp_iqsysmon	Monitors multiple components of Sybase IQ, including the management of buffer cache, memory, threads, locks, I/O functions, and CPU utilization
sp_iqtable	Lists tables and information about them. Omitting the parameter lists all tables in the database. Specifying the <i>table_name</i> parameter lists columns for this table only.
sp_iqtablesize	Gives the size of the specified table
sp_iqtransaction	Shows information about transactions and versions
sp_iqview	Displays views in a database and information about them
sp_iqwho	Displays information about all current users and connections or about a particular user or connection

**Note** Several multiplex stored procedures are also available. See Chapter 10, “System Procedures” in *Sybase IQ Reference Manual* for more information.

## Adaptive Server Enterprise system and catalog procedures

Adaptive Server Enterprise provides system and catalog procedures to carry out many administrative functions and to obtain system information. Sybase IQ has implemented support for some of these procedures.

System procedures are built-in stored procedures used for getting reports from and updating system tables. Catalog stored procedures retrieve information from the system tables in tabular form.

---

**Note** While these procedures perform the same functions as they do in Adaptive Server Enterprise and pre-Version 12 Sybase IQ, they are not identical. If you have preexisting scripts that use these procedures, you may want to examine the procedures. To see the text of a stored procedure, run

```
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'
```

You may need to reset the width of your DBISQL output to see the full text, by clicking Command > Options and entering a new Limit Display Columns value.

---

## Adaptive Server Enterprise system procedures

The following Adaptive Server Enterprise system procedures are provided in Sybase IQ. These stored procedures perform important system management tasks.

System procedure	Description
sp_addgroup	Adds a group to a database
sp_addlogin	Adds a new user account to a database
sp_addrmessage	Adds user-defined messages to SYSUSERMESSAGES for use by stored procedure PRINT and RAISERROR calls
sp_adddtype	Creates a user-defined data type
sp_adduser	Adds a new user to a database
sp_changegroup	Changes a user's group or adds a user to a group
sp_dboption	Displays or changes database options
sp_dropgroup	Drops a group from a database
sp_droplogin	Drops a user from a database
sp_dropmessage	Drops user-defined messages
sp_droptype	Drops a user-defined data type

System procedure	Description
sp_dropuser	Drops a user from a database
sp_getmessage	Retrieves stored message strings from SYSMESSAGES and SYSUSERMESSAGES for PRINT and RAISERROR statements.
sp_helptext	Displays the text of a system procedure or view
sp_password	Adds or changes a password for a user ID

## Adaptive Server Enterprise catalog procedures

Sybase IQ implements all the Adaptive Server Enterprise catalog procedures with the exception of the sp\_column\_privileges procedure. The implemented catalog procedures are described in the following table.

The following list describes the supported Adaptive Server Enterprise catalog procedures.

Catalog procedure	Description
sp_column_privileges	Unsupported
sp_columns	Returns the data types of the specified column
sp_fkeys	Returns foreign key information about the specified table
sp_pkeys	Returns primary key information for a single table
sp_special_columns	Returns the optimal set of columns that uniquely identify a row in a table
sp_sproc_columns	Returns information about a stored procedure's input and return parameters
sp_stored_procedures	Returns information about one or more stored procedures
sp_tables	Returns a list of objects that can appear in a FROM clause

## Catalog stored procedures

In addition to the Adaptive Server Enterprise Catalog stored procedures, there are other system and catalog stored procedures. The following table lists the ones you are most likely to use. For a complete list, see Chapter 10, “System Procedures” in *Sybase IQ Reference Manual*.

Procedure name	Purpose
sp_remote_columns	List remote tables columns and their data types
sp_remote_exported_keys	Get information about remote tables with foreign keys on a specified primary key table
sp_remote_imported_keys	Get information about remote tables with primary keys for a specified foreign key
sp_remote_primary_keys	Get primary key information about remote tables
sp_remote_tables	List tables on a remote server
sp_servercaps	Display information about a remote server's capabilities

## System tables and views

Sybase IQ system tables contain all of the information the database server needs to manage your Sybase IQ system. The system tables reside in the Catalog Store, and are sometimes called catalog tables. For some system tables there are also views that make it easier to display the information in the table. 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 DBISQL options

System tables include:

System table	Description
DUMMY	A table with exactly one row, useful for extracting information from the database
IQ_MPX_INFO	Contains information describing a server in a multiplex database

<b>System table</b>	<b>Description</b>
IQ_MPX_STATUS	Contains dynamic information about a server in a multiplex database
IQ_MPX_VERSIONLIST	Contains one row for each query server in the multiplex.
IQ_SYSTEM_LOGIN_INFO	Stores system defaults for IQ User Administration
IQ_USER_LOGIN_INFO	Stores password expiration, connection limit, and login lock status for each IQ user ID
SYSARTICLE	Describes an article in a SQL Remote publication
SYSARTICLECOL	Describes columns in each article in a SQL Remote publication
SYSCAPABILITY	Each row identifies a capability of a remote server
SYSCAPABILITYNAME	Each row identifies a capability
SYSCHECK	Each row identifies a named check constraint in a table
SYSCOLLATION	Contains the complete collation sequences available to Sybase IQ
SYSCOLLATIONMAPPINGS	Lists the collation sequences available in Sybase IQ and their GPG and JDK mappings
SYSCOLPERM	Records names of individual columns with UPDATE permission
SYSCOLUMN	Describes each column in every table or view
SYSCONSTRAINT	Each row describes a named constraint
SYSDOMAIN	Lists the number, name, ODBC type, and precision of each predefined data type
SYSEVENT	Describes events created with CREATE EVENT
SYSEVENTTYPE	Describes system event types which can be referenced by CREATE EVENT
SYSEXTERNLOGINS	Describes external logins for remote data access
SYSFILE	Lists operating system files and dbspace names for the database

System table	Description
SYSFKCOL	Associates each foreign key column with a primary key column
SYSFOREIGNKEY	Contains general information about each foreign key
SYSGROUP	Describes a many-to-many relationship between groups and members
SYSINDEX	Describes indexes in the database
SYSINFO	Describes database characteristics
SYSIQCOLUMN	Lists information on columns in every table or view in the IQ Store
SYSIQFILE	Lists information on operating system files for the database
SYSIQINDEX	Lists internal information on indexes in the database
SYSIQINFO	Lists additional database characteristics
SYSIQJOININDEX	Describes join indexes in the database
SYSIQJOINIXCOLUMN	Describes columns that participate in join indexes
SYSIQJOINIXTABLE	Lists the tables that participate in each join index in the database
SYSIQTABLE	Describes each table or view in the IQ Store
SYSIXCOL	Describes each index for each column in the database
SYSJAR	Describes each jar file associated with the database
SYSJARCOMPONENT	Describes each jar component associated with the database
SYSJAVACLASS	Contains all information related to Java classes
SYSLOGIN	Lists User Profile names that can connect to the database with an integrated login
SYSOPTION	Lists current SET OPTION settings for all users including the PUBLIC user
SYSPROCEDURE	Describes each procedure in the database

<b>System table</b>	<b>Description</b>
SYSPROCPARM	Describes each parameter to every procedure in the database
SYSPROCPERM	Lists each user granted permission to call each procedure in the database
SYSPUBLICATION	Describes each SQL Remote publication
SYSREMOTEOPTION	Describes the values of SQL Remote message link parameters
SYSREMOTEOPTIONTYPE	Describes the SQL Remote message link parameters
SYSREMOTETYPE	Contains information about SQL Remote
SYSREMOTEUSER	Describes user IDs with REMOTE permissions and the status of their SQL Remote messages
SYSSCHEDULE	Describes the times at which events are to fire
SYSSERVERS	Describes remote servers
SYSSQLSERVERTYPE	Contains information relating to compatibility with Adaptive Server Enterprise
SYSSUBSCRIPTION	Relates each user ID with REMOTE permissions to a publication
SYSTABLE	Describes one table or view in the database
SYSTABLEPERM	Describes permissions granted on each table in the database
SYSTYPEMAP	Contains the compatibility mapping values for SYSSQLSERVERTYPE
SYSUSERMESSAGES	Lists user-defined error messages and their creators
SYSUSERPERM	Lists characteristics of each user ID. Because it contains passwords, you need DBA permissions to select from this table
SYSUSERTYPE	Describes each user-defined data type
SYSWEBSERVICE	Describes web services

System views present the information from their corresponding system tables in a more readable format. In some cases, they omit password information so that they can be accessible to all users. System views include:

<b>System view</b>	<b>Description</b>
SYSARTICLECOLS	Presents a readable version of the table SYSARTICLECOLS
SYSARTICLES	Presents a readable version of the table SYSARTICLES
SYSCAPABILITIES	Presents a readable version of the data from the tables SYSCAPABILITY and SYSCAPABILITYNAM
SYSCATALOG	Lists all tables and views from SYSTABLE
SYSCOLAUTH	Presents column update permission information from SYSCOLPERM
SYSOLUMNS	Presents a readable version of the table SYSCOLUMN
SYSFOREIGNKEYS	Presents foreign key information from SYSFOREIGNKEY and SYSFKCOL
SYSGROUPS	Presents group information from SYSGROUP
SYSINDEXES	Presents index information from SYSINDEX and SYSIXCOL
SYSOPTIONS	Displays option settings contained in the table SYSOPTION
SYSROCAUTH	Presents a readable version of the procedure authorities from SYSUSERPERM
SYSPROCPARMS	Lists all the procedure parameters from SYSPROCPARM
SYSPUBLICATIONS	Presents the user name from SYSUSERPERM for all creators and displays a readable version of the publication name and remarks from SYSPUBLICATION
SYSREMOTEOPTIONS	Presents a readable version of the data from SYSREMOTEOPTION and SYSREMOTEOPTIONTYPE
SYSREMOTETYPES	Presents a readable version of the SQL Remote information from SYSREMOTETYPE
SYSREMOTEUSERS	Lists the information in SYSREMOTEUSER

<b>System view</b>	<b>Description</b>
SYSSUBSCRIPTIONS	Presents a readable version of subscription information from SYSPUBLICATION
SYSTABAUTH	Presents table permission information from SYSTABLEPERM
SYSUSERAUTH	Displays all the information in the table SYSUSERPERM except for user numbers. Because it contains passwords, this system view does not have PUBLIC select permission
SYSUSERLIST	Presents all information in SYSUSERAUTH, except for passwords
SYSUSEROPTIONS	Displays effective permanent option settings for each user
SYSUSERPERMS	Contains exactly the same information as the table SYS.SYSUSERPERM, except the password is omitted
SYSVIEWS	Lists views and their definitions

For a complete description of system tables and views and their contents, see the *Sybase IQ Reference Manual*.

## 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 Adaptive Server Anywhere SQL; the only differences are for certain product capabilities that are supported only for Sybase IQ or for 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 commands and functions you can use. Other chapters of this book tell you about the commands you use to perform various administrative tasks. For complete details of supported commands and functions, see the *Sybase IQ Reference Manual*.

## Types of SQL statements

You use three basic types of SQL 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. They 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 logging

An IQ message log file exists for each database. The default name of this file is *dbname.iqmsg*. The IQ message log file is created when the database is created. Its default location is in the same directory as the Catalog Store, although the database creator may specify another location.

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 start of the file and when you start a database you see output like the following:

```
I. 08/11 16:29:24. 0000000000 OpenDatabase Completed
I. 08/11 16:29:24. 0000000000 IQ cmd line srv opts:
I. 08/11 16:29:24. 0000000000 DB: r/w, Main Buffs=127, Temp Buffs=95,
PgSz=131072/8192blkSz/16bpc
I. 08/11 16:29:24. 0000000000 DB: Frmt#: 23F/2T/1P (FF: 03/18/1999)
I. 08/11 16:29:24. 0000000000 DB: Versn: 12.7.0/040810/P/Mainline/MS/Windows
2000/32bit/2006-08-10 09:54:19
I. 08/11 16:29:24. 0000000000 DB: Name: C:\Program Files\Sybase\ASIQ-
12_7\demo\asiqdemo.db
I. 08/11 16:29:24. 0000000000 DB: Txn ID Seq: 175
I. 08/11 16:29:24. 0000000000 DB: IQ Server fiona_asiqdemo
I. 08/11 16:29:24. 0000000000 DB: Database encryption is OFF.
I. 08/11 16:29:24. 0000000000 Mem: 41mb/M41
    Main      Blks: U4541/80%, Buffers: U4/L0
    Temporary Blks: U65/2%, Buffers: U4/L0
    Main      I: L16/P2 O: C2/D2/P0 D:0 C:100.0
    Temporary I: L28/P0 O: C4/D4/P0 D:0 C: 0.0
```

## Version information in the iqmsg file

Near the beginning of the file you see version information similar to the following:

```
Versn: 12.7/060610/P/GA/Sun_svr4/OS 5.9/64bit/2006-06-10 02:49:46
```

**Table 1-3: Version string elements**

Version string element	Example	Comments
Major release	12.7	First 2 segments indicate major release
Minor release	12.7. <i>n</i>	Third segment indicates minor release
Internal build number	060610	
Internal build type	P	P = Production build
Release type	GA	GA = General Availability ESD = Engineering Software Distribution (Maintenance)
Hardware platform	Sun_svr4	
Operating system version	OS 5.9	OS version and bit mode refer to the system on which the software was built, not where it is running currently
Bit mode	64bit	
Build date and time	2006-06-10 02:49:46	Shown as YYYY-MM-DD hh:mm:ss (ISO datetime format): YYYY 4-digit year MM 2-digit month number (0-12) DD 2-digit day of month number (0-31) hh 2-digit number of complete hours that have passed since midnight (00-23) mm 2-digit number of complete minutes that have passed since the start of the hour (00-59) ss 2-digit number of complete seconds that have passed since the start of the minute (00-59)

## Collation information in the iqmsg file

When you start the database, an entry in the IQ 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 the iqmsg file

The connection handle and database user name, shown in the last line of the output example, are printed after the first transaction begins and before the next transaction starts. This information is printed only once per database connection.

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.

---

**Note** To correlate connection information in the *-zr* log file with that in the *.iqmsg* file, see “Correlating connection information” in the *Sybase IQ Troubleshooting and Recovery Guide*.

---

## Controlling the size of the *iqmsg* file

The message log file continues to exist until you drop the database. By default, the file grows indefinitely. To control the length of this file you can enable message log wrapping. The *IQMSG\_LENGTH\_MB* database option enables message log wrapping, and sets the length of this file. When the file grows to the specified size, new messages are written to the beginning of the file, overwriting existing messages. See Chapter 2, “Database Options” in *Sybase IQ Reference Manual* for details.

Alternatively, you can archive your message log while no users are connected to the database, and then create a new, empty *dbname.iqmsg* file before allowing another user to connect.

## Moving the *iqmsg* file in Sybase Central

When you create a query server, Sybase Central lets you specify the location of its *iqmsg* file if you prefer a default location other than the catalog database directory. In the Create Query Server wizard, on the Temporary DB Information page, select the Override Default File Paths check box. In the Specify the Path to the Message Files pop-up box, type a new path to the *.iqmsg* file.

## Insert notifications in the *iqmsg* file

Insert and load operations are also logged in the IQ message file. For information about insert notification messages logged in the *iqmsg* file, see “Interpreting notification messages” on page 343.

## The utility database

The utility database is essentially a database that never holds data. The database server uses it at times 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.

For more information on the utility database, see “Utility database server security” on page 564.

## Managing very large databases

Sybase IQ’s patented design features permit 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.

## Managing memory use

Allocating memory appropriately is a key factor in performance for all 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 IQ Store and one for the Temporary Store. A local store on a query server shares the main buffer cache. 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.

For complete discussion of Sybase IQ memory use, server and database options that determine your IQ cache sizes, and other options that affect the total available memory on some platforms, see “Managing System Resources,” in *Sybase IQ Performance and Tuning Guide*.

## Managing data loads

As your database grows, it is crucial to manage data loading properly. Means of ensuring that your loads can scale to meet your needs include:

- 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. See “Managing lock contention” on page 489 for details.
- 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 “MAIN\_RESERVED\_DBSPACE\_MB option” and “TEMP\_RESERVED\_DBSPACE\_MB option” in the *Sybase IQ Reference Manual*.

For a list of other features that help you manage load performance, see “Tuning bulk loading of data” on page 387.

## Managing 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 “The process threading model” in *Sybase IQ Performance and Tuning Guide* to understand how Sybase IQ uses threads. See “The database server” in *Sybase IQ Performance and Tuning Guide* to set server options that control thread use.

## Managing disk space

The most important factors in managing disk I/O for an 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, so that you will know when you need to add more space. `sp_iqdbspace` provides additional details.

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. To understand these disk management techniques, see “Balancing I/O” in *Sybase IQ Performance and Tuning Guide*.

## 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. To understand how to best take advantage of this feature, see Chapter 10, “Transactions and Versioning”.

## Creating databases

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 128MB or larger. For more information, see “Choosing an IQ page size” on page 189.

## Creating 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. See Chapter 6, “Using Sybase IQ Indexes” for details on choosing the right indexes.

## Optimizing queries

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.

For information on options that let you examine and influence the query plan, and suggestions for structuring queries for optimal performance, see Chapter 3, “Optimizing Queries and Deletions,” in *Sybase IQ Performance and Tuning Guide*.

## Schema design

Sybase IQ is designed to work best with *denormalized* schemas common in data warehouse design. In a traditional relational database, normalization aids the goals of transaction processing by removing redundancy and improving consistency. In a data warehouse, especially a very large one, you need to be more concerned with the speed of processing queries against enormous amounts of data. For a discussion of denormalization and Sybase IQ, see “Denormalizing for performance” in *Sybase IQ Performance and Tuning Guide*.

## UNION ALL views

Tables with a very large number of rows can take a very long time to load. To resolve this issue, you can use a UNION ALL view. 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 “Using UNION ALL views for faster loads” in *Sybase IQ Performance and Tuning Guide*.

# Running Sybase IQ

## About this chapter

This chapter tells you how to use Sybase IQ to start the database server, start the database, and connect to the database.

## Contents

Topic	Page
Starting the database server	31
Running the server as a Windows service	45
Using command-line switches	45
Monitoring server activity	59
Stopping the database server	62
Restarting multiplex servers with Sybase Central	66
Starting and stopping databases	66
Starting the asiqdemo database	68
Starting and stopping Sybase Central	68

## Starting the database server

Sybase IQ gives you great flexibility in performing these three steps. This chapter explains options for steps 1 and 2 and suggests which to choose, depending on your situation. The next chapter explains options for step 3.

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 utility, `start_asiq`. See “Starting servers with the startup utility” on page 33.

---

**Note** See “Rules for running multiplex servers” on page 34 before starting a multiplex server.

---

- Run the Start Database Server wizard in Sybase Central. See “Starting servers with Sybase Central” on page 35. To start and stop multiplex servers interactively, always use Sybase Central.
- Start the server from the Windows Start menu. See “Starting servers from the Windows Start menu” on page 44.
- Start the server and the sample database with a Sybase-provided configuration file. See “Starting the `asiqdemo` database” on page 68.
- Place a server startup command in a shortcut or desktop icon.

---

**Note** You can also configure Windows systems to start an IQ server automatically when the system is booted. For details, see “Installing Sybase IQ as a Service” in *Sybase IQ Installation and Configuration Guide*.

---

- Include a server startline in an ODBC data source. See “Creating and editing ODBC data sources” on page 97.
- Include a server startline in a utility command.
- Issue a SQL command from Interactive SQL to start an additional server. See “Starting a server from DBISQL” on page 58.

---

**Note** If you will be using remote data access capabilities to insert data from other databases or to issue queries to other databases, see Chapter 16, “Accessing Remote Data” and Chapter 17, “Server Classes for Remote Data Access.”

---

## Starting servers with the startup utility

Sybase recommends that you use the startup utility, start\_asiq or Sybase Central to start servers. This command-line utility runs on all platforms, and ensures that all required parameters are set correctly, except in special situations described later in this chapter.

The general form for the startup utility is as follows:

```
start_asiq [ server-options ] [ database-file  
[ database-options ], ...]
```

The elements of this command line are as follows:

- *server-options* include the database server name and other options that control the behavior of the server, for all databases that are running on that server.
- *database-file* is the file name of the Catalog Store. You can omit this option, or enter one or more database file names on the command line. Each of these databases is loaded and available for applications. If the starting directory contains the database file, you do not need to specify the path; otherwise, you must specify the path. You need not specify the .db file extension.
- *database-options* are options that you can specify for each database file you start, that control certain aspects of its behavior.

In examples throughout this chapter 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).

See “Using command-line switches” on page 45 for a descriptions of configuration files and commonly used options.

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.

---

**Note** 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 “Naming the server and databases” on page 47 for more information on server and database names.

To start the server without starting any database, omit the database file from the `start_asiq` command and specify aservername. 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**

- 1 Change to a writable directory.
- 2 Run the `start_asiq` utility at the system prompt. The simple form of this command is:

```
start_asiq servername [ database ]
```

This command starts the named server as a background process, starts the named database if you specify it, and sets all required startup options. Once the server starts, it sends a message to the window or console where you started the server indicating that the server is running. It also displays other information about your server environment, as well as “possible problems” messages on failure to start. For an example of the version string and other startup messages, see “Message logging” in Chapter 1, “Overview of Sybase IQ System Administration.”

All server messages go to the server log. By default, if `$ASLOGDIR` is defined, the server log is in `$ASLOGDIR/servername.nnn.srvlog`, where *nnn* is the number of times the server has been started. You can also name the server log using the `-o` startup option.

## Rules for running multiplex servers

To start and stop IQ multiplex servers interactively, use Sybase Central. To start IQ multiplex servers in scripts, you may use command line parameters.

Make sure that you follow these rules:

- The IQ Main store is configured for read/write use by only one designated write server. IQ will refuse to start with read/write access unless the connection specifies the designated write server. The database is configured for multiplex operation when the first query server is defined.
- Query servers can only be run on a multiplex database.
- Review and edit the *params.cfg* file in the database directory for the server before starting it. For more about *params.cfg*, see Table 2-1 on page 45.

If you use Sybase Central to start the database:

- Remove any -n switch in a *params.cfg* file used to start a multiplex database.
- The configuration file must be named *params.cfg*.
- The configuration file must be located in the same folder or directory as the database (.DB) file.
- The configuration file must not contain -n to name the server, or the database name or path.

## Starting servers with Sybase Central

Sybase Central, the graphical administration tool, runs on all platforms supported by IQ.

To run Sybase Central, you must configure and run the Sybase IQ Agent, as described in “Running the Sybase IQ Agent” on page 35.

## Running the Sybase IQ Agent

An **agent** is a process that runs on a remote machine and acts on behalf of a client. The Sybase IQ Agent, also called the IQ Agent, enables Sybase Central to:

- Start/stop servers
- Access log files
- Perform system functions

These functions are required by all IQ database administrators. If you have a multiplex database, the IQ Agent must be running *on each machine in your multiplex* in order to fully administer a remote IQ server. Sybase Central can be running while you start and stop IQ Agents.

This section gives instructions on running the IQ Agent on both UNIX and Windows platforms.

### **Specifying the host for the IQ Agent on UNIX**

When you use scripts to start the IQ Agent, use the `-host` parameter to specify the host name explicitly.

To start the agent using the host name returned by `uname -n` use the optional `-host` parameter, as follows:

```
S99SybaseIQAgent12 -host
```

To start the agent using the host's alias enter:

```
S99SybaseIQAgent12 -host <foo>
```

where `foo` is an alias present in the `/etc/hosts` file.

### **Configuring the IQ Agent to start automatically**

#### **Configuring the IQ Agent on UNIX**

To enable automatic startup for the IQ Agent, the system administrator must place the following file in the UNIX startup directory (usually `/etc/rc*`):

```
$ASDIR/bin/S99SybaseIQAgent12
```

After you install Sybase IQ and move the file, the IQ Agent starts automatically whenever you reboot your system.

#### **Configuring the Agent on Windows**

On Windows systems, the installation program sets up the Agent to start automatically each time you reboot the machine. The filename is:

```
SybaseIQagent12.exe
```

#### **Changing the Agent port number**

The IQ Agent port number defaults to 1099. You can override the default value, provided that you do so before the plug-in starts. Changing the default port number lets you to run any number of 12.7 IQ Agents on a given host, or run IQ Agents for Sybase IQ 12.5, 12.6, and 12.7 on the same host. Use the `ASIQPORt` environment variable or the `-port` command line parameter to override the default IQ Agent port number. You can also specify the port per server within Sybase Central, or override the value on the Sybase Central startup command, for example:

```
scjview -DASIQPORt=3356
```

## Overriding the IQ Agent Port Number

Overriding the IQ Agent Port Number on UNIX

On UNIX, you can override the IQ Agent port number on the startup command line; for example:

```
$ASDIR/bin/S99SybaseIQAgent12 -port nnnn
```

Overriding the IQ Agent Port Number on Windows

On Windows, use the Service Manager, as follows:

- 1 Select the name Sybase IQ Agent 12.
- 2 Click the Properties icon or choose Action > Properties from the menu bar.
- 3 Click Stop under Service status to stop the agent.
- 4 In the Start Parameters text box, type `-port nnnn` where nnnn is the port number.
- 5 Click Start to restart the agent.

If the agent fails to start on Windows, check the event log for diagnostic information.

## Setting permissions for the IQ Agent

Setting the permissions on UNIX

The S99SybaseIQAgent12 script should be owned by the same UID that is used to run all the servers. Do not use the root user account.

Setting the permissions on Windows

On Windows, you must change the owner of the Sybase IQ Agent. The user who starts the IQ Agent will be the creator and owner of multiplex databases and server directories, and must have write privileges on all of the servers in the multiplex. By default, the System account starts the Agent.

### ❖ Changing the user

- 1 Choose Administrative Tools > Services.
- 2 Right-click “Sybase IQ Agent” and choose Start.
- 3 On the General tab, choose Automatic for Startup Type.
- 4 On the Log On tab, change the Log on as: option from the System Account to This Account.
- 5 Type *domain\username* for an account with the appropriate privileges in the account text box.

or

Browse to select an account, then click OK.
- 6 Type and reconfirm that account’s password, then click OK.

## Troubleshooting IQ Agent startup

Agent startup on UNIX To check if the Agent is running, run the stop\_asiq utility:

```
stop_asiq -agent
Checking system ...

The following 1 agent(s) are owned by 'ciaran'
## Owner PID Started CPU Time Additional Information
-- -----
- 1: ciaran 6669 Sep.01 5:11 PORT:1100 java -Diq.agent=/work/sybase127/ASIQ-
12_7/java/IQAgent12.jar
-Di q.agent_log=/

-- Do you want to stop the agent displayed above <Y/N>?

Y
```

This output shows that user *ciaran* is running the Agent.

Should you ever receive an error that the Agent is not running, change directory to \$ASDIR/bin, and type S99SybaseIQAgent12 to restart the Agent.

Agent startup on Windows

If you receive an error message that the Sybase IQ Agent is not running, open the Services utility under Control Panel. If IQ Agent does not have the status “Started,” click Start. Restart Sybase Central after starting the Agent.

IQ Agent Log

If you experience any problems with Sybase Central, check the IQ Agent log file.

- If the user explicitly set the IQ Agent port, the log name is:  
%ASDIR%\logfiles\SybaseIQAgent\_pppp.nnn.log where *pppp* is the port number and *nnn* is the number of times you have started the Agent since the directory was last cleaned out.
- If the default IQ Agent port (1099) is used, the log name is:  
%ASDIR%\logfiles\SybaseIQAgent\_nnn.log where *nnn* is the number of times you have started the Agent since the directory was last cleaned out.

On UNIX systems, the file is created in the \$ASLOGDIR directory by default, and running \$ASDIR/bin/S99SybaseIQAgent12 displays a status line with the log file location.

## Running the Start Database Server wizard

### ❖ Starting a non-multiplex server

- 1 Log in using an account with DBA privileges.

2 Start Sybase Central,

To start Sybase Central on UNIX, type:

```
% scjview
```

To start Sybase Central on Windows, run Sybase > Adaptive Server IQ 12 > Sybase Central Java Edition from the Programs menu.

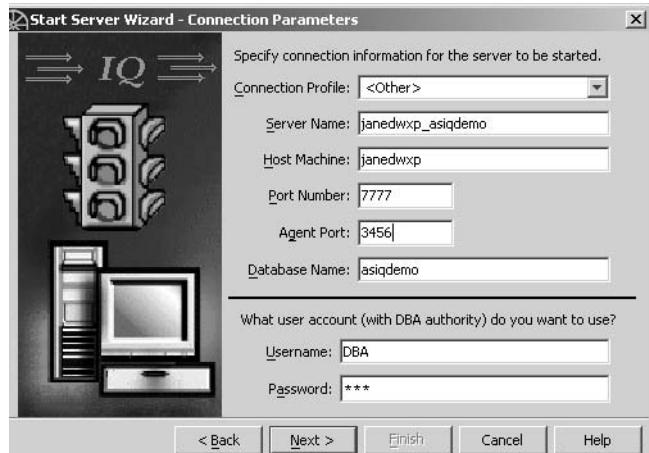
- 3 In the left pane of Sybase Central, select Sybase IQ.
- 4 On the Utilities tab, double-click Start Server.
- 5 Click Next. Starting a single server is the default.
- 6 The information that you specify on the Connection Parameters screen is used each time you start the database.

Connection profiles store parameters to help you connect faster. If a Connection Profile exists, choose it from the dropdown, and click Next.

If no Connection Profile exists for the server, supply connection parameters in the appropriate text boxes. (The wizard creates a Connection Profile from these parameters if you request it on the summary screen.) Do not use connection profiles to start multiplex servers.

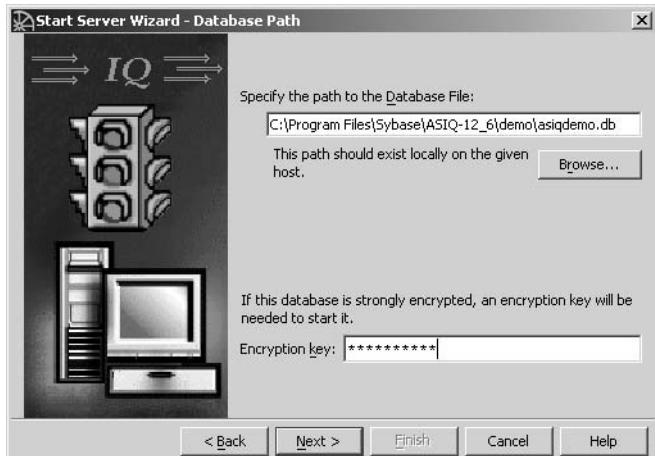
To change default information, simply select it and type over it. In general, the default User ID and password are sufficient; changing the User ID from DBA to another user limits functionality.

Always change the default port number for each server to a different number that is not in use.



- 7 After supplying parameters, click Next.

On the Database Path screen, type the database name and path, or use the Browse key to choose a database on the local host.



If the database is encrypted, type the encryption key in the box.

- 8 Click Next. The Summary screen lists the settings you specified. Server options listed below the line are highlighted if selected, dimmed if not. Click the checkbox to create a connection profile that specifies the parameters in the summary for future connections.



- 9 Click Finish if you are satisfied with options you specified. If not, click the Back button to change information.

❖ **Starting all servers in a multiplex**

An IQ multiplex is a multiserver configuration that supports queries against a shared database from servers with a shared disk subsystem for a common IQ Store. You gain additional CPU power and memory space for processing queries by attaching more systems to the shared disk array.

To use Sybase Central to start multiplex servers interactively.

- 1 Log in using an account with DBA privileges.
- 2 Start Sybase Central,

To start Sybase Central on UNIX, type:

```
% scjview
```

To start Sybase Central on Windows, run Sybase > Adaptive Server IQ 12 > Sybase Central Java Edition from the Programs menu.

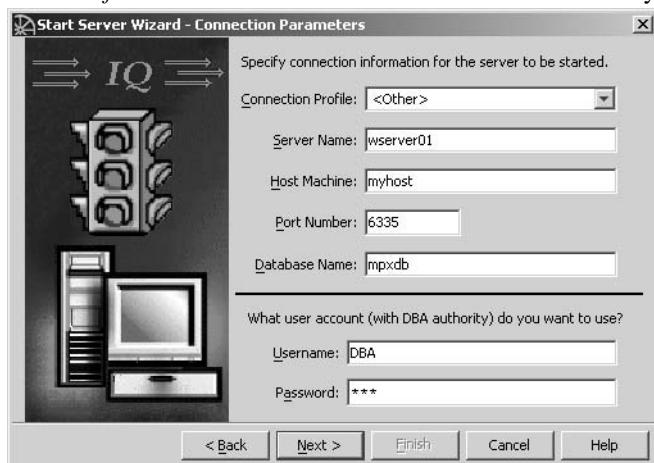
- 3 In the left pane of Sybase Central, select Sybase IQ.
- 4 Select Tools > Sybase IQ > Start Server (ALT+T, I, S)

Updates are made on only one server in a multiplex, the write server. Queries are usually submitted to all servers in the multiplex.

The wizard opens with starting a single server selected by default. Click the “All servers in multiplex” option button to start them all.

- 5 Click Next.

The wizard next displays a series of screens prompting you for server information. *When starting all of the servers in a multiplex, supply information for the write server.* Check all information carefully.



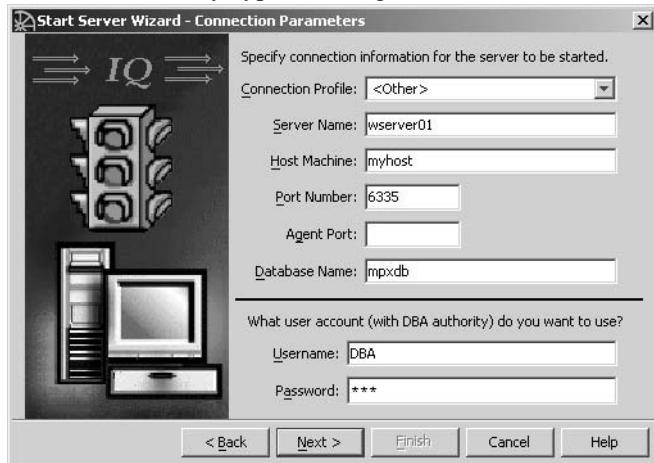
At the bottom of the screen are five command buttons: back, next, finish, cancel and help. These buttons are the same throughout the rest of the wizard and will not be described again.

The information that you specify on this screen is used each time you start the database. To change default information, simply select it and type over it. In general, the default User ID and password are sufficient; changing the User ID from DBA to another user limits functionality.

Always change the default port number for each server to a different number that is not in use.

- 6 Click Next.

If the default IQ Agent port is already in use, the screen displays again with an additional text box where you must specify an IQ Agent port for this connection. If necessary, type an IQ Agent Port and click Next.



If the default IQ Agent Port is not in use, it is used for this connection.

7 Click Next.



- 8 Specify the path to the database file on the write server. You may use the Browse button to locate the file. If there is an encryption key for the database, type it in the space provided.
- 9 Click Next. The next screen summarizes startup settings. Options are listed below the line. They will be highlighted if selected, dimmed if not.




---

**Note** Sybase Central uses the *params.cfg* file to start the server, and requires that this file exist in the directory where the IQ Catalog (.DB file) resides. You may customize this file for each server.

---

- 10 Click Finish if you are satisfied with options you specified. If not, click the Back button to change information.

## Starting servers from the Windows Start menu

This section describes methods for starting the database server that are specific to Windows systems. You can also use any of the generic methods described elsewhere in this chapter.

The easiest way to start the server on Windows is from the Start menu. Select Programs > Sybase > Adaptive Server IQ 12.

From here, you can start the Sybase IQ Demo database, Sybase Central, Interactive SQL Classic, and Interactive SQL Java.

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

## Restarting servers when the Windows host is restarted

Use the 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.

### Synchronizing multiplex servers

Sybase Central provides script files for unattended operations. To create these scripts, open the Multiplex folder, right-click a server name, and choose Generate Admin Scripts. This creates scripts in the database directory for the selected server.

Scripts created on the query server include *sync\_server.bat*, which processes the query server side of a synchronization. This includes copying the database file from the active write server and restarting the query server.

Scripts created on the write server include *sync\_qnode.bat*, which processes the write server side of synchronization. Edit the *sync\_qnode.bat* script for instructions on running these scripts in the proper order. Currently, Sybase does not provide a way to run the *sync\_qnode.bat* script as a service.

## Running 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 *Sybase IQ Installation and Configuration Guide*.

## Using command-line switches

You use command-line switches to define your Sybase IQ environment. *For a complete list of command-line switches and full reference information on them, see Chapter 1, “Running the Database Server” in Sybase IQ Utility Guide.*

Some of the values you can set with command-line options can also be changed with the SET OPTION command. For details of this command and its options, see the *Sybase IQ Reference Manual*.

Displaying command-line options

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

- On UNIX systems, enter:  
`start_asiq -h`
- On Windows systems, enter:  
`start_asiq /?`

Case sensitivity

Command-line switches are case sensitive.

Using 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.

Sybase IQ installs several configuration files.

**Table 2-1: Configuration files**

Filename	Location	Use
<code>asiqdemo.cfg</code>	\$ASDIR/demo (UNIX), %ASDIR%\demo (Windows)	Configuration file for starting the sample database, asiqdemo.

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

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_asiq @mydb.cfg
```

In examples throughout this chapter where there are several command-line switches, 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.

---

**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.

---

Required command-line options

While most of the command-line switches described in Chapter 1, “Running the Database Server” in *Sybase IQ Utility Guide* are optional, you *must* specify the -n switch to run Sybase IQ effectively.

For this release, the *default.cfg* file contains all recommended server startup values.

---

**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 assigns a different one automatically.

Default configuration file

The default configuration file 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\_asiq* command. You can override switches in configuration files by specifying new switches on the *start\_asiq* command line, except for the *-n servername* switch.

Configuration file for the sample database

The *asiqdemo.cfg* file, which you use to start the sample database, sets startup switches to the recommended defaults. This file is installed in *\$ASDIR/demo* on UNIX or *%ASDIR%\demo* on Windows.

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 the *Sybase IQ Reference Manual*.

## Naming the server and databases

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

You may also use the *-n* switch as a database switch (to name the database).

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.

---

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

**Naming databases** 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_asiq -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.

**Naming the server** 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\_sample and the sample database on that server:

```
start_asiq -n Cambridge_sample sample.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. For more information on character sets, see Chapter 11, “International Languages and Character Sets”

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_asiq -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.

---

For information about starting databases on a running server, see “Starting and stopping databases.”

#### 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.

## Controlling performance and memory from the command line

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 the *Sybase IQ Reference Manual* for more information.

Performance tuning suggestions are given throughout this guide. See *Sybase IQ Performance and Tuning Guide* for a full discussion of how Sybase IQ uses memory, disk, and processors, the effect of user connections on resource use, and options you can set to control resource use.

## Setting 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

	<p>The options discussed below, as well as other options you can set once the server is running, determine how much memory is available for these purposes.</p>
IQ buffer cache sizes	<p>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:</p> <ul style="list-style-type: none"><li>• To set buffer cache sizes server-wide for the current server session, specify the server startup options -iqmc (main cache size) and -iqtc (temp cache size). Recommended method.</li><li>• To set cache sizes for a database, you can use the SET OPTION command to set the Main_Cache_Memory_MB and Temp_Cache_Memory_MB database options. This method only lets you set values less than 4GB.</li></ul> <p>In both cases, you must restart the server for the new cache sizes to take effect. If you set IQ buffer cache sizes higher than your system will accommodate, however, Sybase IQ cannot open the database.</p> <p>The server options (-iqmc and -iqtc) override the database option settings. They 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.</p> <p>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.</p>
Concurrent users	<p>Your license sets the absolute number of concurrent users. However, you must also set the -gm switch. This required switch lets you limit the number of concurrent user connections on a particular server.</p> <p>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.</p> <p>On Windows, start_asiq calculates the value of this parameter and sets it using the following formula:</p> <pre>gn_value = gm_value + 5)</pre> <p>Specify a minimum of 25.</p> <p>On UNIX platforms, see the <i>Sybase IQ Installation and Configuration Guide</i> for your platform for more information.</p>

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; see “[Concurrent queries](#).”

#### Concurrent queries

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  $(2 \times \text{the number of CPUs}) + 10$ . 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

The -iqwmem switch creates a pool of “wired” memory on certain UNIX platforms only. For details, see “[Platform-specific memory options](#)” in *Sybase IQ Performance and Tuning Guide*.

---

**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_asiq` command, and the `asiqdemo.cfg` and `default.cfg` configuration files set the -c parameter to 48MB on 64-bit systems and 32MB on 32-bit systems. Sybase recommends using one of these methods.

If you start the server without using `start_asiq`, `asiqdemo.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 *nK* or *nM*.

---

**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. See “IQ buffer cache sizes” on page 50.

---

For more information on setting the Catalog cache size, see the -c, -ca, -ch, and -cl server options in “Starting the database server” in *Sybase IQ Utility Guide*.

## Setting the number of CPUs

The -iqnumbercpus switch on the Sybase IQ startup command lets you specify 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 128.

---

**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 Adaptive Server IQ to a subset of the CPUs within the machine

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

---

## Setting options that affect timing

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.

## Other performance-related options

Several options help you tune network performance. They include -gb (database process priority on Windows), and -p (maximum packet size).

## Controlling permissions from the command line

Starting and stopping databases	Some command-line options control the permissions required to carry out certain global operations.  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:
	<ul style="list-style-type: none"><li>• DBA — Only users with DBA authority can start an extra database.</li><li>• ALL (default in <code>start_asiq</code> and <code>default.cfg</code>) — 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.)</li></ul>

- **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.

---

Creating and deleting 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. See “The utility database” on page 26 for information.

Stopping the server

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.

Loading and unloading 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\_asiq and *default.cfg*) — Any user can load data.
- **NONE** — Data cannot be loaded.

## Setting a maximum Catalog page size

The database server cache is arranged in pages, which are fixed-size areas of memory. Since 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 “Choosing an IQ page size” for more information.

---

## Setting up a client/server environment

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.

See the sections that follow for details.

## Selecting 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.

### Specifying protocols

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_asiq -x "tcpip"
```

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

For detailed descriptions of network communications parameters that can serve as part of the `-x` switch, see “Network communications parameters” on page 157.

## Limiting inactive connections

### Setting the default network timeout

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

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.

---

### Setting the default client timeout

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\_asiq 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.

## Starting a server in forced recovery mode

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. To start the server with these options, see Chapter 2, “System Recovery and Database Repair,” in the *Sybase IQ Troubleshooting and Recovery Guide*.

## Starting a server from DBISQL

If you are already connected to a running database server, you can start a new server from DBISQL. Use the START ENGINE command to start a named server from DBISQL.

---

**Note** 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 STARTLINE, and that environment variables are set appropriately on the system where the server will start.

---

### Example

The following DBISQL command, entered on one line, starts a database server, names it `jill_newsserv`, and specifies the network connection, number of connections, and Catalog page size.

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

## Starting multiple servers or clients on the same machine

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 an Sybase IQ or Adaptive Server Anywhere server is running.* (Interactive SQL will also connect 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 Adaptive Server Anywhere.

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 ASTMP environment variable explicitly on both systems. For details about setting environment variables, see the *Sybase IQ Reference Manual*.
- Create a data source name in the *.odbc.ini* file (on UNIX) for each server and provide detailed connection information. For details, see “Using ODBC data sources on UNIX” on page 105.
- 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 *Sybase IQ Utility Guide*.

## Monitoring server activity

It may be helpful, especially for new users, to monitor server activity. 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 Solaris:

```
Starting server iq_8888 on iqsun at port 8888

Run Directory      : /iqsun1/sybase_127/ASIQ-12_7/demo
Server Executable : /iqsun1/sybase_127/ASIQ-12_7/bin/asiqsvr12
Server Output Log  : /iqsun1/sybase_127/ASIQ-
12_7/logfiles/iq_8888.001.srvlog
Server Version    : 12.7.0/Mainline
Open Client Version : 12.5.1/P
User Parameters   : @asiqdemo.cfg asiqdemo.db
Default Parameters : -c 48m -gc 20 -gd all -gl all -gm 10 -gp 4096 -ti 4400
-gn 25

I. 08/05 18:39:32.      Adaptive Server IQ
I. 08/05 18:39:32.      Version 12.7
I. 08/05 18:39:32.      (64bit mode)
I. 08/05 18:39:32. Copyright 1992-2006 by Sybase, Inc. All rights reserved
I. 08/05 18:39:32.

I. 08/05 18:39:32. Running on SunOS 5.8 Generic_117000-03
I. 08/05 18:39:32. 49152K of memory used for caching
I. 08/05 18:39:32. Minimum cache size: 49152K, maximum cache size: 262008K
I. 08/05 18:39:32. Using a maximum page size of 4096 bytes
I. 08/05 18:39:34. Starting database "asiqdemo" (/iqsun1/sybase_126/ASIQ-
12_7/demo/asiqdemo.db) at Thu Aug 05 2004 18:39
I. 08/05 18:39:35. Transaction log: asiqdemo.log
I. 08/05 18:39:45.      TCPIP link started successfully
I. 08/05 18:39:45. Now accepting requests
New process id is 124
```

Server started successfully

### start\_asiq log file

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

The log file name has this format:

`your_server_name.###.srvlog`

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

```
demo.001.srvlog  demo.002.srvlog
janedemo.001.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.002.srvlog
```

If you don't specify a log file name or IQ cannot find the server name, the log is written to `$ASLOGDIR/start_asiq.log`. If `$ASLOGDIR` is not defined, the log is written to `$ASDIR/logfiles/start_asiq.log`.

When you run `start_asiq`, 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 start_asiq
```

This command returns the commands used to start the various servers, as follows:

```
maryc 24836 25554 0 Feb 09      - 17:36
start_asiq -c 48m -gd all -gl all
-gm 10 -gp 4096 -ti 4400
-tl 300 -iqmt 450
@fnma.cfg asiqdemo.db
janed 28932 38122 0 11:39:24      - 2:10
start_asiq -c 48m -gl all
-gm 10 -gp 4096 -ti 4400
-tl 300 -n janedemo -gd all
-iqmt 256 -x tcpip(port=1872)
```

- Use the `stop_asiq` 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

Use the `-o` parameter on the `start_asiq` startup command to name the server log file, rather than using the default name of `server.###.srvlog`. For example, to save output to a file named `results`, start the server as follows:

```
start_asiq -n imyserver -o results
```

#### 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
```

On UNIX systems, the name of the file that contains stack trace information has this format:

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

## **Stopping the database server**

This section discusses when you need to stop the IQ database server, how to stop it, controlling who can stop it, and stopping the server when you shut down the operating system.

### **When to stop and restart the server**

In a limited set of situations, you need to stop and restart your IQ server:

- 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 “Scope and duration of database options” on page 26 of the *Sybase IQ Reference Manual* for a complete list
- Before closing the operating system session

### **How to stop the server**

The preferred ways to stop the database server are:

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

To shut down servers in an IQ multiplex, open the Multiplex folder, right-click the write server, and choose Server > Stop. You may then choose which server(s) to stop. Click Finish.

- In UNIX, use the stop\_asiq utility at the operating system command line. For details, see “Example — Stop a server with stop\_asiq”.

When you run `stop_asiq`, it displays the following message:

"Please note that 'stop\_asiq' 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 doesn't know how to connect to the server to tell it to shut down. For details on using DBSTOP, see Chapter 1, "Running the Database Server" in the *Sybase IQ Utility Guide*.
- In a DBISQL 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.

- In a UNIX cron or at job, use `stop_asiq` 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_asiq` in such jobs, specify the utility with the appropriate -stop option:

```
stop_asiq -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_asiq -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 pathname to the stop\_asiq executable in the cron statement.

---

**Example — Stop a server with stop\_asiq**

The following example uses the stop\_asiq 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\_asiq 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_asiq
Checking system for ASIQ 12 Servers ...
The following 3 server(s) are owned by other users.

##      Owner      PID      Started    CPU_Time
--  -----  -----  -----  -----
hsin    19895      Mar.21      1:33
start_asiq -c 32m -gd all -gl all -gm 10 -gn 25 -gp 4096 -ti 4400
-n hsin -x tcp
qadaily  24754  01:25:07   1286:53
start_asiq -gn 25 @/exp1/new.cfg asiqdemo.db
-o /exp1/qa
wb    28350      Apr.11      0:20
start_asiq -gn 25 @asiqdemo.cfg -o
/exp1/wb/mysybase12/asiql
```

The following 1 server(s) are owned by 'janed'

```
##      Owner      PID      Started    CPU_Time
--  -----  -----  -----  -----
1:    janed    2838  15:11:37      0:07
start_asiq -c 32m -gd all -gm 10 -gn 25 -gp 4096 -ti 4400 -tl 300
@asiqdemo.cfg
```

```
-- 
Please note that 'stop_asiq' will shutdown a server completely
without regard for users connections or load processes status.
For a finer level of detail the utility 'dbstop' has the options
to control whether a server is stopped 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 (2838) ...
Checkpointing server (2838) ...
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 'janed'
```

#### Example —Stop a server from DBISQL

The following example stops a server from DBISQL:

```
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 DBISQL 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\_asiq 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.

## Shutting down operating system sessions

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. For information on system recovery, see *Sybase IQ Troubleshooting and Recovery Guide*.

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

## Restarting multiplex servers with Sybase Central

If one of the query servers in your multiplex becomes unavailable due to a hardware crash or other failure, you can start it from the right-click menu for that server.

If you need to start all of the servers in the multiplex, use the Start Server wizard for the write server. The writer's Start Server wizard also starts all dbremote processes for all servers selected.

## Starting and stopping databases

You can start databases when you start the server, or after the server is running. To start a database when you start the server, see “Starting the database server” on page 31 for details.

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

Starting a database on a running server

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

- To start a database from DBISQL or Embedded SQL, use the START DATABASE statement. For a description, see START DATABASE statement [DBISQL] in the *Sybase IQ Reference Manual*.

- To start and connect to a database from DBISQL or Sybase Central, use a data source that specifies the database file. See “Working with ODBC data sources” on page 95.
- To start and connect to a database when you start DBISQL from a system command prompt, include the parameter “DBF=*db-file*” in the connection parameters. See “Connecting to the sample database from Sybase Central or DBISQL” on page 86
- To start a database from Sybase Central, see Chapter 4, “Managing Databases,” in *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.

**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.

**Stopping a database**

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 DBISQL or Embedded SQL, use the STOP DATABASE statement.

For information, see STOP DATABASE statement [DBISQL] in the *Sybase IQ Reference Manual*.

## Starting the *asiqdemo* database

You can start the server and the *asiqdemo* database easily, using the configuration file that Sybase IQ provides. This configuration file, called *asiqdemo.cfg*, contains all the parameters necessary to start the sample database.

- ❖ **Starting the server and *asiqdemo* database on UNIX operating systems**
  - From a command line, type the following command:

```
% cd $ASDIR/demo  
% start_asiq @asiqdemo.cfg asiqdemo.db
```

These commands use the configuration file *asiqdemo.cfg* that is created automatically at installation. You can edit this file to change the parameters you use to start the sample database. For example, the server name in this file is *hostname\_asiqdemo*. You can rename the server to a unique name of your choice, like *janed\_server*.

- ❖ **Starting the server and *asiqdemo* database on a Windows system**
  - Click Start on the Taskbar, and select Programs > Sybase > Adaptive Server IQ 12 > Start Sybase IQ Demo Database.

## Starting and stopping Sybase Central

If your system supports a graphical user interface, you will use Sybase Central to perform many administrative tasks. This guide gives summary instructions for using Sybase Central. For more information, see the *Introduction to Sybase IQ*, or use the online help available within Sybase Central.

Starting Sybase Central on UNIX Systems

To start Sybase Central, change directory to *\$\$SYBASE/sybcentral* and type:

```
% scjview
```

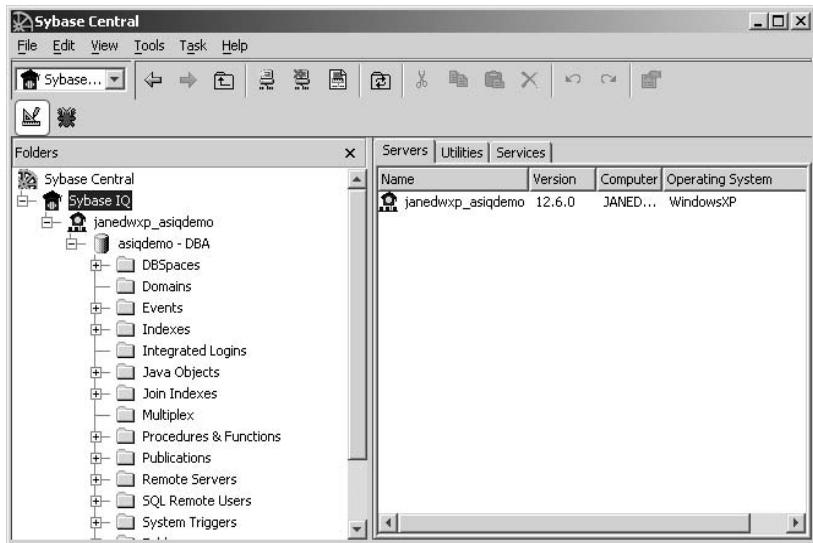
If you have added *\$\$SYBASE/ASIQ-12\_7/bin* or *\$\$SYBASE/bin* to your path, as instructed at the end of the installation, you can issue the *scjview* command from any directory.

Starting Sybase Central on Windows Systems

To start Sybase Central, select Sybase > Adaptive Server IQ 12 > Sybase Central Java Edition from the Programs menu.

The main Sybase Central window appears. It has two panes; the right pane displays the contents of the container selected in the left pane. The first time you open Sybase Central, before connecting to a server, the right pane is empty. To better illustrate the hierarchy, Figure 2-1 shows Sybase Central connected to the demo database.

**Figure 2-1: The Sybase Central Hierarchy**



Plug-ins for Sybase Central, such as the Sybase IQ database management system, occupy the first level in the Sybase Central hierarchy after the root level. A **plug-in** is a graphical tool for managing a particular product. When you install the product, you can also install its Sybase Central plug-in. When you next start Sybase Central, the new product automatically “plugs in” to Sybase Central and appears in the left pane.

## Connecting a plug-in

If you do not see the plug-in for Sybase IQ in the main Sybase Central window, you can connect to it manually.

### ❖ Connecting to a plug-in

- 1 Select Tools > Plug-ins.
- 2 If Sybase IQ is listed, select it and choose Load (Alt+L).

If Sybase IQ is not listed, choose Register (Alt+R). Specify the plug-in jar file option. Then use the Browse button (Alt+r) to find and select the file `$ASDIR/java/IQPlugin12.jar` on UNIX or `%ASDIR%\java\IQPlugin12.jar` on Windows.

- 3 Check Automatically load on startup.
- 4 Click OK.

## **Stopping Sybase Central**

To stop Sybase Central, select File > Exit.

# Sybase IQ Connections

## About this chapter

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 dbisql on a Windows or Linux client to Sybase IQ on a UNIX server.

---

## Contents

Topic	Page
Introduction to connections	72
Connecting from Sybase Central or Interactive SQL	76
Connection shortcuts in Sybase Central	80
Simple connection examples	85
Working with ODBC data sources	95
Creating and editing ODBC data sources	97
Using file data sources	104
Using ODBC data sources on UNIX	105
Connecting to a database using OLE DB	106
Connection parameters	109
How Sybase IQ makes connections	110
Connecting from other databases	117
Testing that a server can be found	118
Using an integrated login	118
Login procedures and IQ User Administration	126
Disconnecting and dropping connections	126
Connection logging	127
Troubleshooting startup, shutdown, and connections	127

## Introduction to connections

This chapter describes how client applications connect to databases. It contains information about connecting to databases from ODBC, OLE DB, and Embedded SQL applications. It also describes connecting from Sybase Central and Interactive SQL.

For more information on connecting to a database from Sybase Open Client applications, see Chapter 15, “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 Appendix B, “Data Access Using JDBC.”

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.

This sounds simple, but some client tools may not clearly indicate connection status, and a failed command is your first indication that the connection does not exist. For a novice user, a quick way to confirm the connection is by a simple `select db_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 connections are established

To establish a connection, the client application calls functions in one of the supported interfaces. Sybase IQ supports the following interfaces:

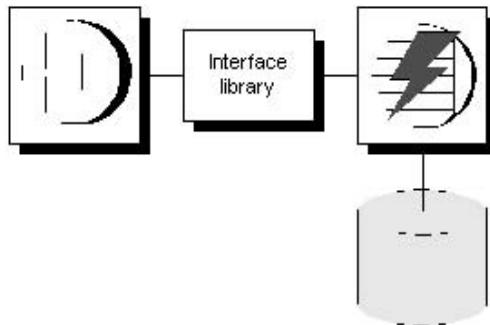
- *ODBC* — ODBC connections are discussed in this chapter.
- *OLE DB* — OLE DB connections are discussed in this chapter.
- *Embedded SQL* — Embedded SQL connections are discussed in this chapter.

- *Sybase Open Client* — Open Client connections are not discussed in this chapter. For information on connecting to IQ from Open Client applications, see Chapter 15, “Sybase IQ as a Data Server”
- *JDBC* — JDBC connections are discussed in this chapter. For more information on connecting via JDBC, see Chapter 4, “Managing Databases,” in *Introduction to Sybase IQ*. To create JDBC data sources, see Appendix B, “Data Access Using JDBC.”

**Note** JDBC provides the link between the execution of Java objects and database operations. For a description of Java support in Sybase IQ, see “Enabling Java in the database” on page 191.

The interface uses connection information included in the call from the client application, perhaps together with information held on disk in a file data source, to locate and connect to a server running the required database. The following figure is a simplified representation of the pieces involved.

**Figure 3-1: Interface library connects applications to servers**



#### Learning about connections

##### If you want ...

An overview of connecting from Sybase Central or Interactive SQL (including a description of the drivers involved)

Some examples to get started quickly

A conceptual overview

To learn what connection parameters are available

To create data sources

##### Consider reading ...

“Connecting from Sybase Central or Interactive SQL” on page 76

“Simple connection examples”

“How connection parameters work”

Chapter 4, “Connection and Communication Parameters”

“Working with ODBC data sources”

If you want ...	Consider reading ...
To see an in-depth description of how connections are established	“Working with ODBC data sources”
To add users and grant them permissions	“How Sybase IQ makes connections”
To diagnose network-specific connection issues	“Troubleshooting network communications” in the <i>Sybase IQ Troubleshooting and Recovery Guide</i>
To learn about character set issues affecting connections	“Connection strings and character sets” on page 532

## 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_asiqdemo;DatabaseName=asiqdemo
```

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)
- DatabaseSwitches (DBS)
- DatabaseName (DBN)

Representing connection strings

This chapter has many examples of connection strings, represented 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.

## Connection parameters are passed as connection strings

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, the following is a typical Collation utility (dbcollat) command line for Windows systems. It should be entered all on one line.

```
dbcollat -c "uid=DBA;pwd=SQL;dbn=asiqdemo"  
c:\temp\asiqdemo.col
```

---

**Note** DBISQL 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 command strings from a command prompt.

---

## Saving 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.

For more information on ODBC data sources, see “Working with ODBC data sources” on page 95.

## **Connecting from Sybase Central or Interactive SQL**

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.

For connection examples, including examples for Sybase Central and Interactive SQL, see “Simple connection examples” on page 85.

---

**Note** To avoid ambiguity, specify connection parameters for DBISQL 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*. For a complete list, see Chapter 4, “Connection and Communication Parameters.”.

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

---

## Opening the Connect dialog

A common Connect dialog is available in both Sybase Central and Interactive SQL to let you connect to a database.

When you start Sybase Central, you need to manually display this dialog.

When you start Interactive SQL, the dialog automatically appears; you can also make it appear for a new connection by choosing File > New Window.

❖ **Opening the Connect dialog (Sybase Central)**

- In Sybase Central, choose Tools > Connect.

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.

---

**Tip**

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

❖ **Opening the Connect dialog (Interactive SQL)**

- 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, you connect to the Sybase IQ sample database by specifying Database file: *asiqdemo.db* (in *\$ASDIR/demo* or *%ASDIR%\demo*) 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 that the box “Search network for database servers,” on the Database tab is checked.

---

**Note** When you connect to a user-created database, you must fill out both the Database File and Database Name fields. Supply the entire pathname in the Database Name field.

---

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

## Specifying a driver for your connection

When you work with a database, all your requests and commands go through a driver to the database itself. Interactive SQL and Sybase Central support two main JDBC drivers: Sybase jConnect, and the iAnywhere JDBC Driver. Both are included with Sybase IQ.

Sybase jConnect is a platform-independent, pure Java, JDBC driver. It is enabled by default for the IQ plug-in to Sybase Central, and as an option on Interactive SQL (Java).

The iAnywhere JDBC Driver is enabled by default on Interactive SQL (Java). It provides JDBC 2.0 support and fully scrollable cursors, which the jConnect 5.5 driver does not.

As you connect to a database in the Connect dialog, you can choose which driver you want to use for the connection, or rely on the default. Both drivers perform well for most users. You may find that one driver or the other works best for you.

For more information on JDBC drivers including required software, see “Choosing a JDBC driver” on page 788, “Using the Sybase jConnect JDBC driver” on page 808, and “Working with ODBC data sources” on page 95.

## Data sources and the jConnect driver

As a general rule, the jConnect driver cannot use ODBC data sources.

However, Sybase Central and Interactive SQL (Java) are special cases. When you use the jConnect driver with either of them, you can specify an ODBC data source to establish a connection.

This customized functionality is only available while you are working in Sybase Central or Interactive SQL. If you are constructing a JDBC application, do not try to use an ODBC data source to connect to a database.

### ❖ Specifying a driver for the connection

- 1 In Interactive SQL (Java), choose SQL > Connect to open the Connect dialog.
- 2 Configure the necessary settings on the Identification and Database tabs of the log.
- 3 On the Advanced tab of the dialog, select either jConnect 5 or iAnywhere JDBC Driver, or rely on the default: jConnect for Sybase Central, iAnywhere JDBC for Interactive SQL (Java).

## Working with 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 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.

## Connection shortcuts in Sybase Central

By following the procedures in this section, you can simplify Sybase Central connections.

### Creating server objects

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

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

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

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 DSEdit. 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.

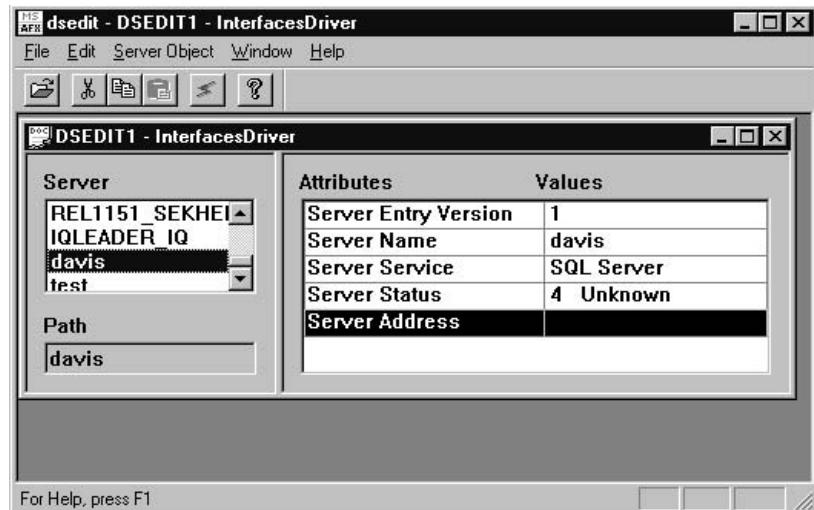
❖ **Adding a server object**

- 1 Open the Utilities folder in Sybase Central and double-click Directory Services Editor.

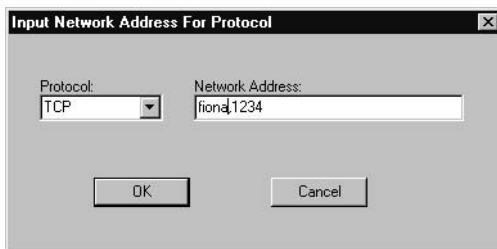


- 2 Click OK.
- 3 Select Server Object > Add from the menu bar.
- 4 Type the server name. *The server name in your DSEdit entry must be the same as the database name.* Click OK when finished.

5 Select Server Address.



- 6 Right-click on Server Address and select Modify Attribute.
- 7 Click Add.
- 8 Use the dropdown box to select the desired protocol. For example, TCP.
- 9 In the Network Address box, type the hostname and port number, separated by a comma, for example:



You can substitute the network address for the hostname (for example, "157.133.75.47,1234").

- 10 Click OK.
- 11 To test the server's availability right-click on any of the Attributes and select Ping Server.
- 12 Click Ping to test the server.
- 13 Click OK and Done when finished.

Object names allow all users at your installation to connect more quickly. They can simply click the Server Name dropdown to choose from all servers defined and supply their own usernames and passwords to connect.

## Creating connection profiles

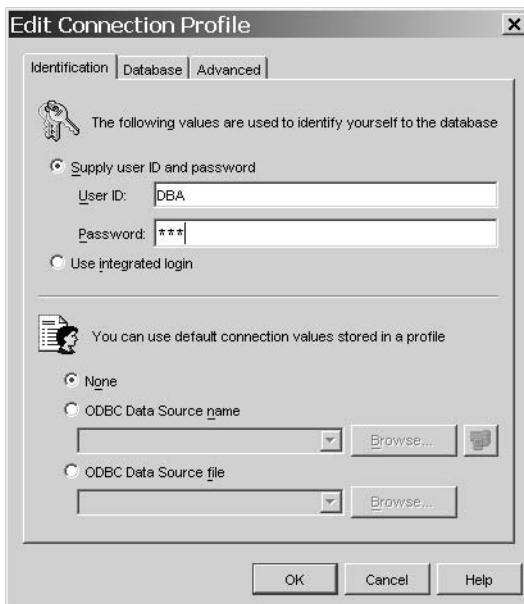
Connection profiles make it easy to connect to databases automatically when an individual user boots a system, or to connect without typing connection parameters. Because connection profiles include the username and password, they are better suited to individual use than across an entire installation. 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.

### ❖ Creating a connection profile

- 1 From the Sybase Central menu, choose Tools > Connection Profiles or F9.
- 2 Click New (Alt+N).

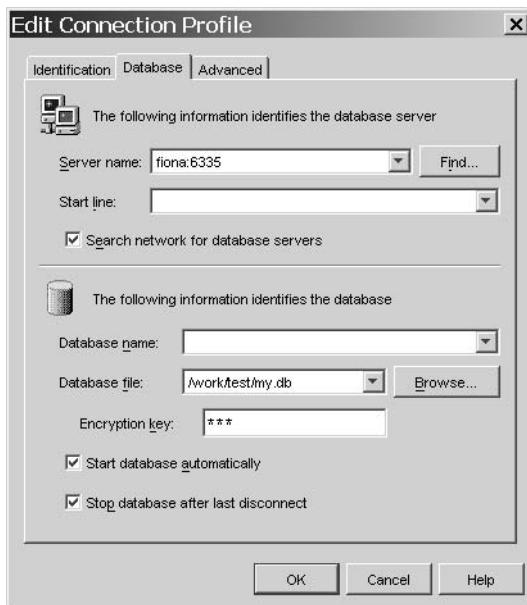


- 3 Type a name for the profile and click OK.



- 4 On the Identification tab, type the User ID and Password. Do not click OK yet.

- 5 On the Database tab, type the Host name and Port number in the Server name field or select a server from the Server Name pulldown.



- 6 Click OK.
- 7 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.
- 8 If you choose not to connect automatically on startup, you can now connect from Sybase Central by simply choosing Tools > 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, this section may be all you need, for example, if you are running the asiqdemo sample database on a local server and are not connected to a network. However, in most IQ environments, in order to ensure that you can connect and disconnect properly, a very complete set of connection parameters is essential.

For steps in connecting to a database using Sybase Central, see the *Introduction to Sybase IQ*. For more detailed information on available connection parameters and their use, see “Connection parameters” on page 109.

## Connecting to the sample database from Sybase Central or DBISQL

Many examples and exercises throughout the documentation start by connecting to the sample database from Interactive SQL, also called DBISQL.

### ❖ Connecting to the sample database (Sybase Central)

- 1 Start Sybase Central as appropriate for your system.

*On UNIX*, you must source the *ASIQ-12\_7.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:

```
% $ASDIR/bin/S99SybaseIQAgent12
```

To start Sybase Central, type:

```
% cd $SYBASE/sybcentral  
% scjview
```

---

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

---

*On Windows*, to start Sybase Central, choose Programs > Sybase > Adaptive Server IQ 12 > Sybase Central Java Edition.

- 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 asiqdemo server from the Find Server screen and click OK.
- ❖ **Connecting to the sample database (Interactive SQL)**
- 1 To start Interactive SQL from the Start menu, choose Programs > Sybase > Adaptive Server IQ 12 > Interactive SQL Java.
  - 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 DBISQL, see Chapter 2, “Using Interactive SQL (dbisql),” in the *Sybase IQ Utility Guide*.

## Connecting to a database on your own machine from Sybase Central or Sybase IQ

The simplest connection scenario is when the database you want to connect to resides on your own machine. If this is the case for you, ask yourself the following questions:

- 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.

The procedures below depend on your answers to the questions.

- ❖ **Connecting to a database on an already-running local server**
- 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.

❖ **Connecting to a database that is not yet running**

- 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).

---

**Tip**

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” on page 77
- “Simple connection examples” on page 85

## Connecting to other databases from DBISQL

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

❖ **Connecting to a database from DBISQL on UNIX**

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

```
start_asicq dbname
```

- 2 Start DBISQL by typing at a system command prompt:

```
DBISQL -c "uid=userID;pwd=password" -host hostname  
-port portnum
```

The -c parameter specifies connection parameters. See “Connection parameters” for more about connection parameters.

For example, to connect to the sample database on remote host *fiona*, you enter:

```
dbisql -c "uid=DBA;pwd=SQL" -host fiona -port 1870
```

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

## Connecting using command line utilities

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

### ❖ Connecting from a UNIX system

- 1 Make sure that your PATH and other environment variables are correctly set, as described in Chapter 1, “File Locations and Installation Settings” in the *Sybase IQ Reference Manual*.
- 2 To ensure that the sample database is loaded on a running server, at the UNIX prompt enter:

```
ps -eaf | grep asiqdemo
```

If you need to start the sample database, enter:

```
cd $ASDIR/demo  
start_asiq @asiqdemo.cfg asiqdemo.db
```

- 3 Start DBISQL by entering the following command:

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

Make sure you replace *servername* with the same server name that was supplied in the *start\_asiq* 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 foreign host, you must add the host name and port number. For example:

```
dbisql -c "uid=DBA;pwd=SQL;eng=SERV1_asiqdemo;
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

- 1 Choose Sybase > Adaptive Server IQ 12 > Interactive SQL Java from the Programs menu, or at the Windows command prompt enter

```
dbisql
```

---

**Note** If you prefer the C version of Interactive SQL, choose Sybase > Adaptive Server IQ 12 > Interactive SQL Classic from the Programs menu, or at the Windows command prompt enter

```
dbisqlc
```

---

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 DBISQL connect dialog appears.

- 2 In the CONNECT dialog, enter your user name and password.  
For example, for the `asiqdemo` 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_asiqdemo`” for the `asiqdemo` database). This name must be unique on your local area network.

For remote servers, specify the server as *host name:port number*.

The default port number is 2638, but if the host was started with a different number, use that instead. You can find the port number by running Sybase > ODBC Administrator. Select the Data Source on the User DSN tab, then click Configure. The port number is listed on the Network tab. You can also 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 DBISQL window appears. The DBISQL window displays the database name, user ID, and server name for the connection on its title bar.

If you connect using DBISQLC, the words “Connected to database” appear in the Statistics window along with a message displaying the collation used by the database.

## Connecting to an embedded database

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.

### Using the DBF 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 sample database as an embedded database:

```
dbf=path\asiqdemo.db  
uid=DBA  
pwd=SQL
```

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

**Using the StartLine (Start) parameter**

The following connection parameters show how you can customize the startup of the sample database as an embedded database. This is useful if you wish to use command-line options, such as the cache size:

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

**Example: connecting from DBISQL**

In this example, the sample database is an embedded database within DBISQL.

❖ **Connecting to an embedded database from DBISQL in Windows**

- 1 Start DBISQL with no databases running. You can use either of the following ways:
  - From the Windows Programs menu, choose Sybase > Adaptive Server IQ 12 > Interactive SQL Java.
  - Type `dbisql` at a system command prompt.
- When DBISQL 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 sample database in the Database File field. For example, if your installation directory is `c:\sybase\ASIQ-12_7` you should enter the following:

```
c:\sybase\ASIQ-12_7\asiqdemo.db
```

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

## Connecting using a data source

You can save sets of connection parameters in a data source. ODBC and Embedded SQL applications like DBISQLC use data sources. You can create data sources from the ODBC Administrator. See “Creating and editing ODBC data sources” for details.

If you are constructing an application, you should only use data sources for ODBC applications. It is possible to specify data sources when you are using the JDBC driver (jConnect), but only within Sybase Central or Interactive SQL. For more information, see “Specifying a driver for your connection” on page 78.

- ❖ **Connecting from Sybase Central or Interactive SQL using a 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 Adaptive Server IQ Demo data source holds a set of connection parameters, including the database file and a Start parameter to start the database.

---

**Note** You can also specify the data source name by including the `dsn` connection parameter in the `dbisql` command, as follows:

```
dbisql -c "dsn=Adaptive Server IQ Demo"
```

---

The asiqdemo data source

The asiqdemo data source holds a set of connection parameters, including the database file and a Start parameter to start the sample database. The server name in this data source is “*hostname\_asiqdemo*” where *hostname* represents your system name.

## Using 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 Adaptive Server 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
```

No 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
```

For more information about default behavior, see “How Sybase IQ makes connections”.

## 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. They follow the procedure outlined in “How Sybase IQ makes connections” when connecting to a database.

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 there are any). For example, the following command starts the collation utility on the sample database on the default server, using the user ID DBA and the password SQL and the *asiqdemo.col* collation file:

```
dbcollat -c "uid=DBA;pwd=SQL;dbn=asiqdemo"  
c:\temp\asiqdemo.col
```

- 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 Chapter 1, “File Locations and Installation Settings” in *Sybase IQ Reference Manual*.
- 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 Chapter 3, “Database Administration Utilities” in the *Sybase IQ Utility Guide*.

## Working with 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.

## Where data sources are held

Although data sources are especially designed for Windows, Sybase IQ allows you to create and use them on UNIX servers as well. This allows ODBC-based client applications to connect to databases on UNIX servers.

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.

Sybase IQ permits ODBC data sources for several purposes in addition to Windows applications using the ODBC interface:

- Sybase IQ client applications on UNIX can use ODBC data sources, as well as those on Windows operating systems.
- Sybase IQ client applications using the OLE DB or Embedded SQL interfaces can use ODBC data sources, as well as ODBC applications.
- Interactive SQL and Sybase Central can use ODBC data sources.

## Creating and editing ODBC data sources

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

To create ODBC data sources on UNIX systems, see also “Using ODBC data sources on UNIX”.

### 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.

For more information, see “Simple connection examples” on page 85 and “Assembling a list of connection parameters” on page 112.

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

### Creating a data source from the ODBC Administrator

#### ❖ **Creating an ODBC Data Source (ODBC Administrator)**

1 Start the ODBC Administrator:

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

or

Programs > Sybase > ODBC Administrator

The ODBC Data Source Administrator appears.

2 Click Add.

The Create New Data Source wizard appears.

3 Select the Sybase IQ 12 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.

For details of the ODBC configuration box and its tabs, see “Configuring ODBC data sources” on page 99

---

**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.

### ❖ Creating an ODBC data source (Command line)

- 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 sample database. The command must be entered on one line:

```
iqdsn -w "My DSN"  
"uid=DBA;pwd=SQL;dbf=c:\Program Files\Sybase\ASIQ-  
12_7\demo\asiqdemo.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 “Data Source utility (*iqdsn*)” in *Sybase IQ Utility Guide*.

#### ❖ Testing an ODBC Data source

- 1 Start the database. For example, to start the Sample Database, select Sybase > Adaptive Server IQ 12 > Start Sybase IQ Demo Database from the Programs menu.
- 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 pathnames.

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

If you need to access Windows across a network in order to create an ODBC data source, see the *Sybase IQ Installation and Configuration Guide*.

## Configuring ODBC data sources

This section describes the meaning of each of the options on the ODBC configuration dialog, organized by tab.

### 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.

For more information, see the DataSourceName connection parameter in the *Sybase IQ Reference Manual*.

**Description** You can enter an optional longer description of the Data Source.

**Translator** Choose Adaptive Server Anywhere 7.0 Translator if your database uses an OEM code page. If your database uses an ANSI code page, which is the default, leave this unchecked.

**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.

For more information, see Chapter 10, “Transactions and Versioning.”

**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 SQLStatistics.

**Delphi applications** Check this box to improve performance for Borland Delphi applications. When this option is checked, one bookmark value is assigned to each row, instead of the two that are otherwise assigned (one for fetching forwards and a different one for fetching backwards).

Delphi cannot handle multiple bookmark values for a row. If the option is unchecked, scrollable cursor performance can suffer since scrolling must always take place from the beginning of the cursor to the row requested in order to get the correct bookmark value.

**Suppress fetch warnings** Check this box to suppress warning messages that are returned from the database server on a fetch.

Sybase IQ version 12 and later return a wider range of fetch warnings than earlier versions of the software. For applications that are deployed with an earlier version of the software, you can select this option to ensure that fetch warnings are handled properly.

**Prevent Driver not capable errors** The Sybase IQ 12 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.

**Delay AutoCommit until statement close** Check this box if you wish the Sybase IQ 12 ODBC driver to delay the commit operation until a statement has been closed.

**Describe cursor behavior** Select how often you wish a cursor to be re-described when a procedure is executed or resumed.

**Test Connection** Tests if the information provided will result in a proper connection. In order for the test to work a user ID and password must have been specified.

## Login tab

**Use integrated login** Connects using an integrated login. The User ID and password do not need to be specified. Instead, your operating system user id and password are supplied to the Sybase IQ integrated login mechanism. To use this type of login users must have been granted integrated login permission. The database being connected to must also be set up to accept integrated logins. Only users with DBA access can administer integrated login permissions.

For more information, see “Using integrated logins” on page 119.

**User ID** Provide a place for you to enter the User ID for the connection.

For more information, see “Userid connection parameter [UID]” on page 156.

**Password** Provides a place for you to enter the password for the connection.

For more information, see “Password connection parameter [PWD]” on page 153.

**Encrypt password** Check this box if you wish the password to be stored in encrypted form in the profile.

For more information, see “EncryptedPassword connection parameter [ENP]” on page 146.

## Database tab

**Server name** Provides a place for you to enter the name of the IQ server.

For more information, see “EngineName connection parameter [ENG]” on page 146.

**Start line** Enter the server that should be started. Only provide a Start Line parameter the database server you are connecting to is not currently running. For example:

```
C:\Program Files\Sybase\ASIQ-12_7\win32\start_asiq.exe  
-gm 10 -gp 4096 -c 32M  
dbf=path\asiqdemo.db  
uid=DBA pwd=SQL
```

For more information, see “StartLine connection parameter [START]” on page 156.

**Database name** Provides a place for you to enter the name of the Sybase IQ database that you wish to connect to.

For more information, see “[DatabaseName connection parameter \[DBN\]](#)” on page 143.

**Database file** Provides a place for you to enter the full path and name of the Sybase IQ database file on the server machine. You can also click Browse to locate the file. For example:

```
C:\Program Files\Sybase\ASIQ-12_7\demo\asiqdemo.db
```

For more information, see “[DatabaseFile connection parameter \[DBF\]](#)” on page 141.

**Encryption key** If your database is strongly encrypted, you must supply the correct encryption key to access any part of the database.

For more information, see “[Encryption connection parameter \[ENC\]](#)” on page 147.

**Start database automatically** Causes the database to start automatically (if it is not already running) when you start a new session. This option is enabled when you supply a database file or start line.

**Stop database after last disconnect** Causes the automatic shutdown of the server after the last user has disconnected.

## Network tab

**Select the network protocol and options** These check boxes specifies what protocol or protocols the ODBC DSN will use to access a network database server. In the adjacent boxes, you may enter communication parameters that establish and tune connections from your client application to a database.

- **TCP/IP** If you want to use ECC\_TLS (Certicom) or RSA\_TLS encryption for network packets, you must select the TCP/IP protocol to access the network database server.

For example, you could enter the following communication parameters for a TCP/IP connection HOST=my\_server;PORT=4563.

- **Named pipes** The Named Pipes protocol is used for client/server communication on the same machine. If you want to run under a certified security environment, you can use the Named Pipes protocol. It is only provided on supported Windows platforms.
- **Shared memory** The shared memory protocol is used for communication between a client and server running under the same operating system on the same machine.

For a TCP/IP example, see “Creating an ODBC Data Source (ODBC Administrator)” on page 97.

For more information see the CommLinks connection parameter, and Network communications parameters, in Chapter 4, “Connection and Communication Parameters.”

**Liveness timeout** A liveness packet is sent across a client/server to confirm that a connection is intact. If the client runs for the liveness timeout period without detecting a liveness packet, the communication will be severed. This parameter works only with network server and TCP/IP communications protocols. The default is 120 seconds.

For more information, see “LivenessTimeout connection parameter [LTO]” on page 152.

**Idle timeout** Set the amount of client idle time before the connection is terminated. If a client runs for the idle timeout period without submitting a request, the connection is severed.

The default client idle time is set by the -ti server option, whose default setting in start\_asiq is 4400 minutes.

For more information, see “Idle connection parameter [IDLE]” on page 149.

**Buffer size** Set the maximum size of communication packets, in bytes. The default buffer size is 1460.

For more information, see “CommBufferSize connection parameter [CBSIZE]” on page 138.

**Select the method for encryption of network packets** Allows the encryption of packets transmitted from the client machine over the network.

- **None** Communication packets transmitted from the client are not encrypted. This is the default setting.
- **Simple** Communication packets transmitted from the client are encrypted with simple encryption. Simple encryption is supported on all platforms, as well as on previous versions of Sybase IQ.
- **ECC TLS** Communication packets transmitted from the client are encrypted with ECC\_TLS (formerly Certicom) encryption. This is a type of strong encryption. It is only available over the TCP/IP protocol.  
In the adjacent field you must supply a trusted certificate value.
- **RSA TLS** Communication packets transmitted from the client are encrypted with RSA\_TLS encryption. This is a type of strong encryption. It is only available over the TCP/IP protocol.

In the adjacent field you must supply a trusted certificate value.

For more information, see “Encryption connection parameter [ENC]” on page 147

## Advanced tab

This tab provides advanced fields that are for special purposes and are not required by the majority of users.

**Connection name** The name of the connection that is being created.

**Character set** Lets you specify a character set (a set of 256 letters, numbers, and symbols specific to a country or language). The ANSI character set is used by Microsoft Windows. An OEM character set is any character set except the ANSI character set.

**Allow multiple record fetching** Enables multiple records to be retrieved at one time instead of individually. By default, multiple record fetching is allowed.

**Display debugging information in a log file** The name of the file in which the debugging information is to be saved.

**Additional connection parameters** Enter any additional options here. Parameters set throughout the remainder of this dialog take precedence over parameters typed here.

# Using 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

❖ **Creating an ODBC file data source (ODBC Administrator)**

- 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.

Creating a file data source using a text editor

A file data source is a text file, so it can be edited using 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 = asiqdemo
DBA = DBA
PWD = SQL
```

## Using ODBC data sources on UNIX

On UNIX operating systems, ODBC data sources are held in a file named *.odbc.ini*. When creating a *.odbc.ini* file on any UNIX system, you must use the long form of each identifier, for example:

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

Password=SQL

You can enter any connection parameter in the *.odbc.ini* file. Network communications parameters are added as part of the CommLinks (LINKS) parameter. For a complete list of network parameters, see “Network communications parameters” on page 157.

You can create and manage ODBC data sources on UNIX using the iqdsn command-line utility. For more information see “Creating an ODBC data source from the command line” on page 98.

File location

References to ODBC functions are resolved at run time. The client looks for the *.odbc.ini* file in the paths specified by the following variables (in order):

- 1   ODBCINI environment
  - 2   ODBCHOME
- 

**Note** On UNIX systems, Sybase IQ installs only the ODBC driver, and not the driver manager. The name of the driver file includes an operating system-specific extension, for example, *so* for Solaris systems. For example, on a Sun Solaris system, 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/ASIQ-12\_7/lib/libdbodbc9.so.1*. If you are creating a custom ODBC application, you can link directly to *dbodbc9.so*.

If Sybase IQ does not detect the presence of an ODBC driver manager, it will use *~/.odbc.ini* for data source information. Otherwise, it will query the driver manager for data source information.

See Chapter 7, “ODBC Programming” in the *Adaptive Server Anywhere Programming Guide* to ensure that you are using the correct driver for your platform.

---

## Connecting to a database 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 *Adaptive Server Anywhere Programming Guide*.

---

**Note** Sybase IQ support for certain features used with OLE DB differs from the support of Adaptive Server 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 ASA OLE DB provider** The Adaptive Server 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 ASAProv.

When the ASAProv 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 ASAProv provider is called. If you change the location of your DLL, you must reregister it.

If you use the Adaptive Server Anywhere OLE DB provider, ODBC is not required in your deployment.

For more information about OLE DB providers, see *Adaptive Server Anywhere Programming Guide*.

- **Microsoft OLE DB provider for ODBC** Microsoft provides an OLE DB provider with a short name of MSDASQL.

The MSDASQL provider makes ODBC data sources appear as OLE DB data sources. It requires the Sybase IQ ODBC driver.

## Connecting 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.

For a list of connection parameters, see “Connection parameters” on page 109.

- The Open method initiates a connection.

For more information about ADO, see *Adaptive Server Anywhere Programming Guide*.

### Example

The following Visual Basic code initiates an OLE DB connection to Sybase IQ:

```
' Declare the connection object
Dim myConn as New ADODB.Connection
myConn.Provider = "ASAProv"
myConn.ConnectionString = "Data Source=Sybase IQ Demo"
myConn.Open
```

## Connection parameters

Chapter 4, “Connection and Communication Parameters” describes Sybase IQ connection parameters.

## Connection parameter tips

Connection parameters often provide more than one way of accomplishing a given task. This is particularly the case with embedded databases, where the connection string starts a database server. For example, if your connection starts a database, you can specify the database name using the DatabaseName (DBN) connection parameter or using the DatabaseSwitch (DBS) parameter.

Here are some recommendations and notes for situations where connection parameters conflict:

- **Specify database files using DBF** You can specify a database file on the StartLine (START) parameter or using the DatabaseFile (DBF) connection parameter (recommended).
- **Specify database names using DBN** You can specify a database name on the StartLine (START) parameter, the DatabaseSwitch (DBS) connection parameter, or using the DatabaseName (DBN) connection parameter (recommended).
- **Use the Start parameter to specify cache size** Even though you use the DatabaseFile (DBF) connection parameter to specify a database file, you may still want to tune the way in which it starts. You can use the StartLine (START) connection parameter to do this. For recommended parameters, see “Required command-line options” on page 46.

For example, you may need to provide additional Catalog cache memory on the StartLine (START) connection parameter, as in the following sample set of embedded database connection parameters, where the default Catalog cache size for Windows, 32M, is increased to 48M:

```
DBF=path\asiqdemo.db
DBN=Sybase IQ Demo
ENG=Sample Server
UID=DBA
PWD=SQL
Start=start_asiq -c 48M -gd all -gp 4096
```

## How Sybase IQ makes connections

Who needs to read this section?

This section describes how the interface libraries establish connections.

In many cases, establishing a connection to a database is straightforward using the information presented in the preceding sections of this chapter. However, if you are having problems establishing connections to a server, you may need to understand how Sybase IQ establishes connections in order to resolve your problems.

For more information on resolving connection problems, see *Sybase IQ Troubleshooting and Recovery Guide*.

---

**Note** If you have no problem establishing connections to your database, you do not need to read this section.

---

The software follows exactly the same procedure for each of the following types of client application:

- Any ODBC application using the SQLDriverConnect function, which is the common method of connection for ODBC applications. Many application development systems, such as Sybase PowerBuilder and Power++, belong to this class of application.
- Any client application using Embedded SQL and using the recommended function for connecting to a database (`db_string_connect`).

The SQL CONNECT statement is available for Embedded SQL applications and in Interactive SQL. It has two forms: CONNECT AS... and CONNECT USING... . All the database administration tools, including utilities and Interactive SQL, use `db_string_connect`.

## Steps in establishing a connection

To establish a connection to Sybase IQ, the client application carries out the following steps:

- 1 *Locate the interface library.* The client application must locate the ODBC driver or Embedded SQL interface library.
- 2 *Assemble a list of connection parameters.* Since connection parameters may be provided in several places, such as data sources, a connection string assembled by the application, and an environment variable, Sybase IQ assembles the parameters into a single list.

- 3 *Locate a server.* Using the connection parameters, Sybase IQ locates a database server on your machine or over a network.
- 4 *Locate the database.* Once it locates the server, Sybase IQ locates the database you are connecting to.

The following sections describe each of these steps in detail.

## Locating the interface library

The client application makes a call to one of the Sybase IQ interface libraries. In general, the location of this DLL or shared library is transparent to the user. Here we describe how the library is located, in case of problems.

### ODBC driver location

For ODBC, the interface library is also called an ODBC driver. An ODBC client application calls the ODBC driver manager, and the driver manager locates Sybase IQ's driver.

The ODBC driver manager looks in the supplied data source in the *odbc.ini* file or registry to locate the driver. When you create a data source using the ODBC Administrator, Sybase IQ fills in the current location for your ODBC driver.

### Embedded SQL interface library location

Embedded SQL applications call the interface library by name. The name of the Sybase IQ Embedded SQL interface library is as follows:

- *Windows: dblib9.dll*
- *UNIX: libdblib9* with an operating system-specific extension.

The locations that are searched depend on the operating system:

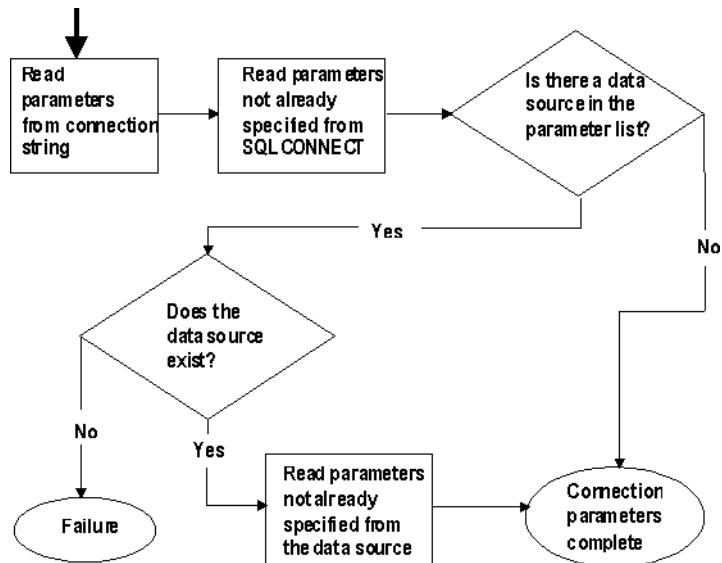
- On Windows, the client application looks for files in the current directory, in the system path, and in the *Windows* and *Windows\system* directories.
- On UNIX, the client application looks for files in the system path and the user library path.

### When the library is located

Once it locates the interface library, the client application passes a connection string to it. The interface library uses the connection string to assemble a list of connection parameters, which it uses to establish a connection to a server.

## Assembling a list of connection parameters

The following figure illustrates how the interface libraries assemble the list of connection parameters they will use to establish a connection.



### Notes

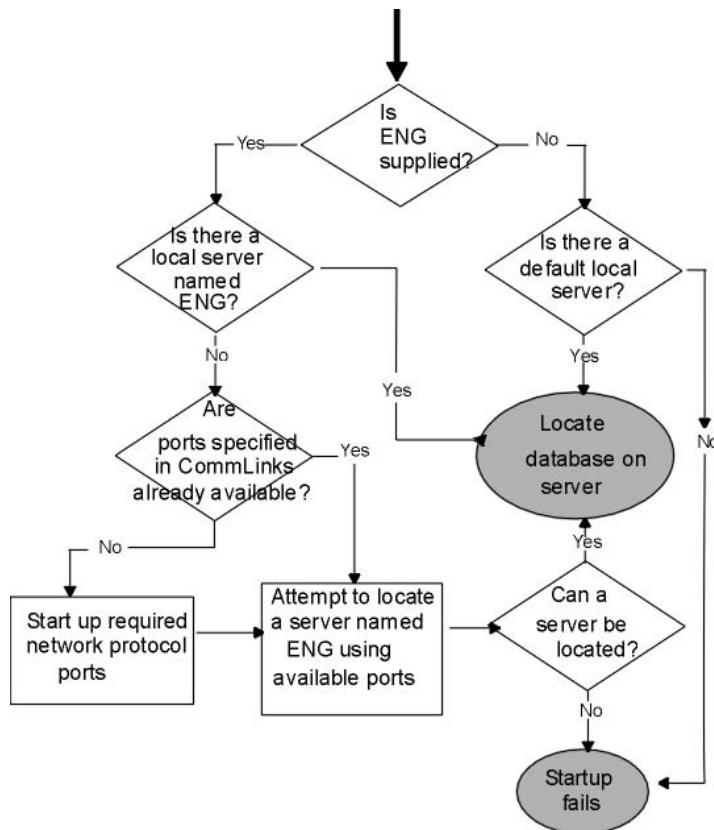
Key points from the figure are as follows:

- *Precedence* — Parameters held in more than one place are subject to the following order of precedence:
  - Connection string
  - SQLCONNECT
  - Data source
 That is, if a parameter is supplied both in a data source and in a connection string, the connection string value overrides the data source value.
- *Failure* — Failure at this stage occurs only if you specify in the connection string or in SQLCONNECT a data source that does not exist in the client connection file.
- *Common parameters* — Depending on other connections already in use, some connection parameters may be ignored, including:
  - **AutoStop** Ignored if the database is already loaded.
  - **DatabaseFile** Ignored if DatabaseName is specified and a database with this name is already running.

The interface library uses the completed list of connection parameters to attempt to connect.

## Locating a server

In the next step towards establishing a connection, Sybase IQ attempts to locate a server. If the connection parameter list includes a server name (EngineName (ENG) connection parameter), Sybase IQ searches first for a database server of that name, and then searches over a network. If no ENG parameter is supplied, Sybase IQ looks for a default server.



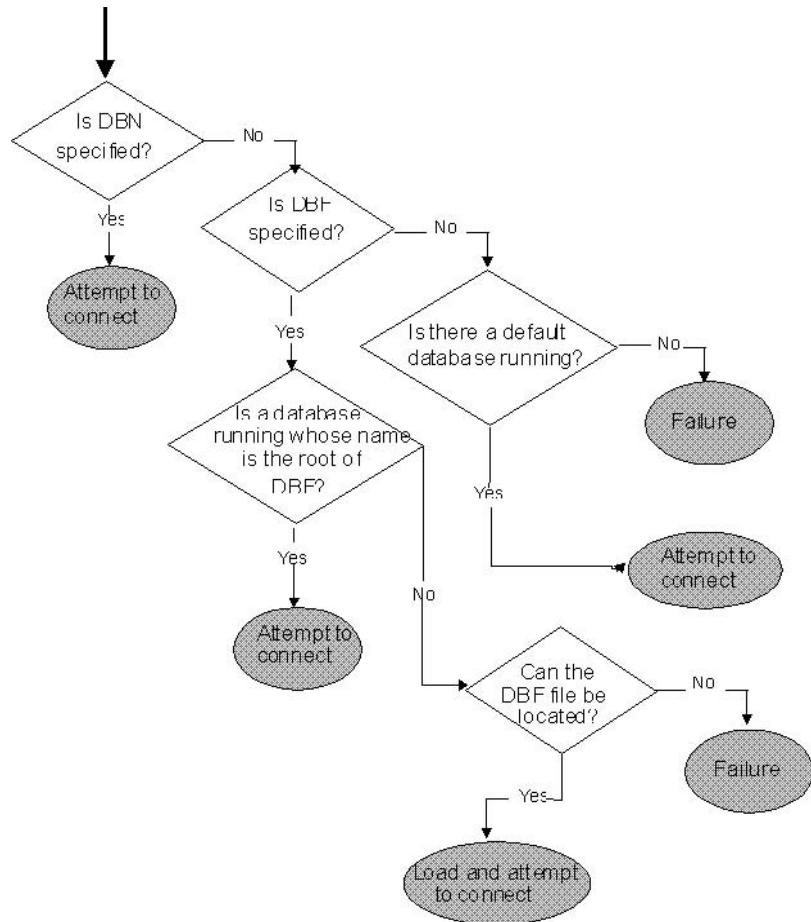
If Sybase IQ locates a server, it tries to locate or load the required database on that server. For information, see “Locating the database” on page 115.

### Notes

- For local connections, locating a server is simple. For connections over a network, you can use the CommLinks (LINKS) parameter to tune the search in many ways by supplying network communication parameters.
- The network search involves a search over one or more of the protocols that Sybase IQ supports. For each protocol, the network library starts a single **port**. All connections over that protocol at any one time use a single port.
- You can specify a set of network communication parameters for each network port in the argument to the CommLinks (LINKS) parameter. Since these parameters are used only when the port first starts, the interface library ignores any connection parameters specified in CommLinks (LINKS) for a port already started.
- Each attempt to locate a server (the local attempt and the attempt for each network port) involves two steps. First, Sybase IQ looks in the server name cache to see if a server of that name is available. Second, it uses the available connection parameters to attempt a connection.
- The default server is the first one started on a machine. This server gains use of the shared memory port.

## Locating the database

If the interfaces library successfully locates a server, it then tries to locate the database. For example:



### Notes

- If you rely on the DBF parameter, the DBF path must either be an absolute path, or relative to where the server was started, in order for Sybase IQ to find the database it specifies. For example, if you specify `./foo/asiqdemo`, it looks in the directory above where the server is, and then in `foo`.
- The default database is the one started with the server.

## Server name caching for faster connections

When the DoBroadcast (DOBROAD) communication parameter is set to DIRECT or ALL, the network library looks for a database server on a network by broadcasting over the network using the CommLinks (LINKS) connection parameter.

### Tuning the broadcast

The CommLinks (LINKS) parameter takes as argument a string that lists the protocols to use and, optionally for each protocol, a variety of network communication parameters that tune the broadcast.

For a complete listing of network communications parameters, see Chapter 4, “Connection and Communication Parameters.”

### Caching server information

Broadcasting over large networks to search for a server of a specific name can be time-consuming. Leaving the DoBroadcast (DOBROAD) communication parameter at the default value (ALL) speeds up network connections by saving the protocol the first connection to a server was found on, and its address, to a file, and using that information for subsequent connections.

The server information is saved in a file named *asasrv.ini*, in your Sybase IQ executable directory. The file contains a set of sections, each of the following form:

```
[Server_name]
Link=protocol_name
Address=address_string
```

### How the cache is used

When a connection specifies a server name, and a server with that name is not found, the network library looks first in the server name cache to see if the server is known. If there is an entry for the server name, an attempt is made to connect using the link and address in the cache. If the server is located using this method, the connection is much faster, as no broadcast is involved.

If the server is not located using cached information, the connection string information and CommLinks parameter are used to search for the server using a broadcast. If the broadcast is successful, the server name entry in the named cache is overwritten.

---

**Note** If a server name is held in the cache, the cache entry is used before the CommLinks string.

---

## Interactive SQL connections

The Interactive SQL (DBISQL) utility has a different behavior from the default Embedded SQL behavior when a CONNECT statement is issued while already connected to a database. If no database or server is specified in the CONNECT statement, Interactive SQL connects to the current database, rather than to the default database. This behavior is required for database restarting operations.

For an example, see “CONNECT statement [ESQL] [DBISQL]” in the *Sybase IQ Reference Manual*.

## Connecting 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 related information, see “Matching Adaptive Server Enterprise data types” on page 382. 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 *Sybase IQ Installation and Configuration Guide* for your platform.
- Access data in Sybase IQ and Adaptive Server 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.

## Testing that a server can be found

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
```

For more information on dbping options, see the *Sybase IQ Utility Guide*.

For more information about printing more output during connection attempts, see “LogFile connection parameter [LOG]” on page 153. This feature is especially useful in conjunction with the Logfile connection parameter, described in the same chapter.

## Using an integrated login

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. This section describes the integrated login feature.

### 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 an integrated login

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.

---

For more information about security and integrated logins, see “Security concerns: unrestricted database access”.

## Using integrated logins

Several steps must be implemented in order to connect successfully via an integrated login.

### ❖ Using 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).

- **Standard** This is the default setting, which does not permit integrated logins. An error occurs if an integrated login connection is attempted.
- **Mixed** With this setting, both integrated logins and standard logins are allowed.
- **Integrated** 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).

---

### Example

The following SQL statement sets the value of the LOGIN\_MODE database option to Mixed, allowing both standard and integrated login connections:

```
SET OPTION "PUBLIC".LOGIN_MODE = Mixed
```

## Creating an integrated login

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.

A user profile does not have to exist for it to be mapped to a database user ID. More than one user profile can be mapped to the same user ID.

Only users with DBA authority are able to create or remove an integrated login mapping.

An integrated login mapping is made either using a wizard in Sybase Central or a SQL statement.

❖ **Mapping an integrated login (Sybase Central)**

- 1 Connect to a database as a user with DBA authority.
- 2 Open the Integrated Logins folder for the database, and double-click Add Integrated Login. 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.

❖ **Mapping an integrated login (SQL)**

- 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 GRANT statement in *Sybase IQ Reference Manual*.

## Revoking integrated login permission

You can remove an integrated login mapping using Interactive SQL.

❖ **Revoking an integrated login permission (SQL)**

- 1 Connect to a database with DBA authority.
- 2 Execute a REVOKE INTEGRATED LOGIN FROM statement.

### Example

The following SQL statement removes integrated login permission from the Windows user pchin.

```
REVOKE INTEGRATED LOGIN  
FROM pchin
```

For more information, see REVOKE statement in *Sybase IQ Reference Manual*.

## 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.

### Interactive SQL Examples

For example, a connection attempt using the following Interactive SQL statement will succeed, providing the user has logged on with a user profile name that matches a integrated login mapping in a default database of a server:

```
CONNECT USING 'INTEGRATED=yes'
```

The following DBISQL statement.

```
CONNECT
```

can connect to a database if all the following are true:

- A server is currently running.
- The default database on the current server is enabled to accept integrated login connections.
- An integrated login mapping has been created that matches the current user's user profile name.
- If the user is prompted with a dialog box by the server for more connection information (such as occurs when using the DBISQL utility), the user clicks OK *without* providing more information.

### Integrated logins via ODBC

A client application connecting to a database via ODBC can use an integrated login by including the Integrated parameter among other attributes in its Data Source configuration.

Setting the attribute 'Integrated=yes' in an ODBC data source causes database connection attempts using that DSN to attempt an integrated login. If the LOGIN\_MODE database option is set to Standard, the ODBC driver prompts the user for a database user ID and password.

## Security concerns: unrestricted database access

The integrated login features works by using the login control system of Windows in place of the system 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.

---

## Setting temporary public options for 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.

For more information on using the SET OPTION statement see Chapter 6, “SQL Statements” in *Sybase IQ Reference Manual*.

## 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.

## Creating a default integrated login user

A default integrated login user ID can be created so that connecting via an integrated login will be successful 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 will successfully map 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.

## **Login procedures and IQ User Administration**

The IQ User Administration facility helps you manage users and connections to a database. Using these system procedures, DBAs can add or drop users, limit connections for a user or a database, set password expirations, and lock out users.

When users log in, the `sp_iq_process_login` system procedure checks the `IQ_USER_LOGIN_INFO_TABLE` to ensure that the user is permitted to log in. It checks that the current number of connections for that user and for the database is not exceeded, that the user password is not expired, and that the user is not locked out. The database option `LOGIN_PROCEDURE`, which specifies a procedure to run when a user attempts to connect an IQ database, is set to `sp_iq_process_login` by default.

You can also create your own stored procedure that runs when users connect to an IQ database. This procedure can control initial conditions for users, such as option settings for users. It can also be used to disable new connections, based on criteria you define.

Set the `LOGIN_PROCEDURE` database option to the procedure you want to call when users log in. For details, see the *Sybase IQ Reference Manual*.

## **Disconnecting and dropping connections**

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 Java, a user selects `SQL > Disconnect` (or `Command→Disconnect` in Interactive SQL Classic)
- In Sybase Central, a user selects `Tools > Disconnect`
- An application with active connections, such as DBISQL or Sybase Central, is closed

Users with DBA authority can also drop a specific connection:

- In DBISQL or Embedded SQL, by issuing a `DROP CONNECTION` statement

- In Sybase Central, by accessing the Connected Users folder for the database

For more about displaying connection IDs and active connection information, see “sp\_iqconnection procedure” in *Sybase IQ Reference Manual*. See *Sybase IQ Reference Manual* Chapter 6, “SQL Statements” for the DISCONNECT and DROP CONNECTION statements. For information on Sybase Central, see Chapter 4, “Managing Databases” in *Introduction to Sybase IQ*.

## 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 “LOG\_CONNECT option” on page 105 in the *Sybase IQ Reference Manual*.

## Troubleshooting startup, shutdown, and connections

See the sections that follow for help in resolving problems with your database server, connections, and DBISQL. For other troubleshooting hints, including network communication issues, see the *Sybase IQ Troubleshooting and Recovery Guide*.

### What to do if you can't start Sybase IQ

This section describes some common problems when starting the database server.

## Ensure that your files are valid

The server will not start if the existing transaction log is invalid. For example, during development, you may replace a database file with a new version, without deleting the transaction log at the same time. This causes the transaction log file to be different from the database, and results in an invalid transaction log file.

The same is true for the IQ Temporary Store file.

## Ensure that you have sufficient disk space for your temporary file

Sybase IQ uses a temporary file to store information while running. This file is stored in the directory pointed to by the TMP or TEMP environment variable, or the ASTMP environment variable on UNIX. On Windows, this directory is typically *c:\temp*. On UNIX, if more than one database server is running on the same machine, each user needs a separate temporary directory; typically this directory is set to */tmp/.userid*.

If you do not have sufficient disk space available to the temporary directory, you will have problems starting the server.

## Ensure that network communication software is running

Appropriate network communication software must be installed and running before you run the database server. If you are running reliable network software with just one network installed, this should be straightforward. If you experience problems, if you are running non-standard software, or if you are running multiple networks, read the discussion of network communication issues in the *Sybase IQ Installation and Configuration Guide*. You should also refer to the documentation of your network vendor.

You should confirm that other software that requires network communications is working properly before running the database server.

For example, if you are running under the TCP/IP protocol, you may want to confirm that ping and telnet are working properly. The ping and telnet applications are provided with many TCP/IP protocol stacks.

If you are using NetBIOS under Windows, you may want to confirm that the *chat* or *winpopup* application is working properly between machines running client and database server software.

## Check environment variables

In order to start the server, certain environment variables must be set properly. On Windows, the installation procedure sets any environment variables you need automatically. On UNIX, you must set these variables. While it is unlikely for these settings to change, you should check to be sure they are correct if you cannot start the server. See Chapter 1, “File Locations and Installation Settings” in *Sybase IQ Reference Manual* for information on environment variable settings.

## Debugging network communications startup problems

If you are having problems establishing a connection across a network, you can use debugging options at both client and server to diagnose problems. On the server, you use the -z command-line option. The startup information appears on the server window (or the server log, if you start the server with start\_asiq). You can use the -o *filename* option to log the results to an output file if you start the server with start\_asiq.

## What to do if you can't connect to a database

If you are unable to connect to a Sybase IQ database, check the items described below.

- Check that you have entered your data source name correctly, or that you have selected the correct server name for a JDBC connection.
- Check that your data source (DSN or FILEDSN) contains the correct server name, database, network parameters, and any other connection information you expect.
  - On Windows, select Settings > Control Panel > Data Sources (ODBC), then select the User DSN tab and select the source name for the server you want.
  - On UNIX, check your *.odbc.ini* file
- Check that the database server you want is running.

- On Windows: If the server is running on the local host, check the icon on the toolbar at the bottom of the screen. If the server is remote, select Settings > Control Panel > Data Sources (ODBC) to open the ODBC Administrator, then select the User DSN tab, highlight the source name for the server you want, and click Configure > Test Connection. If that server is running, and the data source is set correctly, you see a Connection Successful message.
- On UNIX, enter the following at the system prompt, substituting the name of your database server for *asiqdemo*:

```
ps -eaf | grep asiqdemo
```

- Check that any connection parameters you enter on the command line are correct.

The Interactive SQL utility behaves differently from Embedded SQL when a CONNECT statement is issued while already connected to a database. If no database or server is specified in the CONNECT statement, Interactive SQL connects to the current database, rather than to the default database. This behavior is required for database reloading operations.

If you specify a database without a server name, DBISQL attempts to connect to the specified database on the current server. (Note that you must specify the database name defined in the -n database switch, not the database file name.) If you specify a server name without a database name, DBISQL connects to the default database on the specified server.

- If you are connecting to a database that is not running, check that the server was started with the -gd ALL switch. If not, then only the DBA can start databases on that server, by first connecting to the utility\_db database and then issuing a START DATABASE command for the desired database.
- Check that you have permission to use the database for the requested operation. The DBA or database owner must grant you CONNECT permission; see “Managing individual user IDs and permissions”.
- If you are having problems establishing a connection across a network, you can use debugging options at both client and server to diagnose problems. On the server, you use the -z command-line option. The startup information appears on the server window: you can use the -o option to log the results to an output file.

- Check that all of the files exist for the database you have requested. At a minimum, there must be an IQ Store (*dbname.iq*), a Catalog Store (*dbname.db*), an IQ Temporary Store (*dbname.iqttmp*), a transaction log (*dbname.log* This may be missing if the database is newly created and has not been modified.), and a message file (*dbname.iqmsg*). The names shown here in parentheses are the default format; yours may be different.
- Check that any restores have completed successfully.

See also “How Sybase IQ makes connections”.

## Stopping a database server in an emergency (UNIX)

Always try first to stop the server using the methods described in “Stopping the database server” on page 62. If you are unable to stop it using those methods, and if you started the database server as a batch or background process (using `start_asiq`), try the following:

- 1 If possible, you should make sure that no users are connected to the database.
- 2 At the UNIX prompt, enter the following command:

```
kill -hup pid
```

where *pid* is the process id of the database server you are stopping.

See also Chapter 1, “Troubleshooting Hints,” in the *Sybase IQ Troubleshooting and Recovery Guide* for important information on shutting down your server if it fails to shut down normally, and on cleaning up your system after killing a process.

## Using the UNIX kill command to stop DBISQL

Normally, you use the File > Exit command to exit from DBISQL. If in an emergency you need to stop DBISQL from the UNIX command level, make sure you stop the right process. When you start DBISQL, two processes are started: a shell process running the `dbisql` executable, and the Java executable process that runs from the `$$SYBASE/ASIQ-12_7/jre/bin/native_threads` directory. The Java executable is the process you need to kill.

Use a command like the following to find both processes:

```
ps -ef | egrep "jre|java|dbisql"
ami 1712 21442 0 12:01:17 pts/11 0:00 /bin/sh /mysystem/ami/ASIQ-
```

```
12_7/bin/dbisql
ami 1748 21894 0 12:10:13 pts/14    0:00 egrep jre|java|dbisql
ami 1715 1712 0 12:01:18 pts/11    0:07
/mysystem/ami/ASIQ-12_7/jre/bin/.../bin/sparc/native_threa
```

Then use a command like `kill -9` to kill the process id for the Java process, which has *jre* in the path, for example:

```
kill -9 1715
```

## Resolving problems with your DBISQL window on UNIX

The interactive DBISQL utility on UNIX uses character-based windows. These windows rely on line-drawing characters that are part of the ASCII character set, and some other character sets as well. The ability of DBISQL to display these windows correctly depends on the type of terminal you use, and the character set translation your operating system uses. If your windows appear to be drawn with accented characters rather than line-draw characters, you can still use them to enter commands and receive output as described throughout this book.

DBISQL uses function keys for many operations. Some UNIX windowing environments may not support these function keys, unless you make some adjustments.

For help in improving the appearance of DBISQL windows, or if you are unable to use function keys in DBISQL, see Chapter 2, “Using Interactive SQL (dbisql),” in *Sybase IQ Utility Guide*.

# Connection and Communication Parameters

## About this chapter

This chapter provides a reference for connection parameters used to establish and describe connections from client applications to a database.

## Contents

Topic	Page
Connection parameters	133
Network communications parameters	157

## Connection parameters

This section describes each connection parameter. After each parameter name, the short form is listed in square brackets. You can use the short form as an abbreviation in connect commands.

Connection parameters are included in connection strings. They can be entered in the following places:

- In an application's connection string
- In an ODBC data source
- In the Sybase IQ Connect dialog

For more information see Chapter 3, “Sybase IQ Connections.”

The ODBC configuration dialog and the Sybase IQ Connect dialog for Windows operating systems share a common format. Some of the 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. For more information on embedded databases, see “Simple connection examples” on page 85.
- **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 “Ping utility (dbping)” in *Sybase IQ Utility Guide*.

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. For more information on the character set issues, see “Connection strings and character sets”.

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.

## ApplInfo connection parameter [App]

Function

To help administrators identify the origin of particular client connections from a database server.

Usage

Anywhere

Default	Empty string
Description	<p>This connection parameter is sent to the database server from Embedded SQL, ODBC, or OLE DB clients, as well as Interactive SQL (Classic on Windows and 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.</p> <p>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:</p> <pre>SELECT connection_property( 'AppInfo' )</pre> <p>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=<i>value</i> pairs. The valid keys are as follows:</p> <ul style="list-style-type: none"><li>• <b>API</b> DBLIB, ODBC, OLEDB, or ADO.NET (ODBC is returned of the iAnywhere JDBC driver).</li><li>• <b>APPINFO</b> If you specified AppInfo in the connection string, the string entered</li><li>• <b>EXE</b> The name of the client executable (Windows only)</li><li>• <b>HOST</b> The host name of the client machine</li><li>• <b>IP</b> The IP address of the client machine (UNIX only)</li><li>• <b>OS</b> The operating system name and version number (for example, Sun Solaris 2.9)</li><li>• <b>PID</b> The process ID of the client</li><li>• <b>THREAD</b> The thread ID of the client</li><li>• <b>VERSION</b> The version of the connection protocol in use, including major and minor values, and a build number (for example 9.0.1.1549)</li><li>• <b>TIMEZONEADJUSTMENT</b> The number of minutes that must be added to the Coordinated Universal Time (UTC) to display time local to the connection.</li></ul> <p>If you specify a debug log file in your client connection parameters, the APPINFO string is added to the file.</p>
Examples	<ul style="list-style-type: none"><li>• Connect to the sample database from Interactive SQL (the iAnywhere JDBC driver is used by default):<pre>dbisql -c "uid=DBA;pwd=SQL;dbf=C:\Program</pre></li></ul>

```
Files\Sybase\ASIQ-12_7\demo\asiqdemo.db"
```

View the application information:

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

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

```
HOST=machine-name;
OS=Windows XP Build 2600 Service Pack 1;
PID=0x724;
THREAD=0x6bc;
EXE=C:\Program Files\Sybase\ASIQ-
12_7\win32\dbisqlg.exe;
VERSION=9.0.2.1021;
API=ODBC;
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]

Function	To force each statement to COMMIT before execution.
Usage	ODBC
Default	NO
Description	By default, statements issue a COMMIT <i>after</i> 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\ASIQ-
12_7\demo\asiqdemo.db
AutoPreCommit=Y
UserID=DBA
Password=SQL
```

## AutoStart connection parameter [Astart]

Function	To prevent 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	<ul style="list-style-type: none"><li>The following data source fragment prevents a database server from being started if no network server is located:</li></ul>

```
[My Sample Database]
DatabaseFile=c:\sybase\ASIQ-12_7\demo\asaiqdemo.db
Autostart=No
UserID=DBA
ENG=network_server
```

## AutoStop connection parameter [Astop]

Function	To prevent 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\ASIQ-12_7\demo\asiqdemo.db
Autostop=No
UserID=DBA
```

## **CharSet connection parameter [CS]**

Function	To specify the character set to be used on this connection.
Usage	Anywhere
Default	The locale character set. For information on how this is determined, see “Determining locale information” on page 540.
Description	If you supply a value for CharSet, the specified character set is used for the current connection.  For a list of valid character set values, see “Character set labels” on page 519.
See also	“How connection parameters work” in <i>Sybase IQ System Administration Guide</i>

## **CommBufferSize connection parameter [CBSIZE]**

Function	To set the maximum size of communication packets, in bytes.
Usage	Anywhere
Values	Integer

Default	If no CommBufferSize value is set, the CommBufferSize is controlled by the setting on the server, which defaults to 1460 bytes.
Description	<p>The CommBufferSize parameter specifies the size of communications packets, in bytes. The minimum value of CommBufferSize is 300, and the maximum is 16000.</p> <p>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.</p> <p>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.</p> <p>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.</p> <p>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.</p>
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.

## CommLinks connection parameter [Links]

Function	To specify client side network communications links.
Usage	Anywhere
Values	<i>String</i>
Default	Use only the shared memory communications link to connect.
See also	“Network communications parameters” on page 157

“-x list” in Table 1-1, *Sybase IQ Utility Guide*

Description	<p>If you specify <code>CommLinks=ALL</code>, the client searches for a server using all available communication protocols. Since there may be an impact on performance if you specify <code>CommLinks=ALL</code>, use this setting only when you don't know which protocol to use.</p> <p>If you specify one or more protocols in the <code>CommLinks</code> (LINKS) connection parameter, the client uses the named communication protocol(s), <i>in the order specified</i>, 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.</p> <p>If you do not specify a <code>CommLinks</code> (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 <code>CommLinks=ShMem</code>. The shared memory protocol is used for communication between a client and server running under the same operating system on the same machine.</p>
-------------	--

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. For a list of parameters, see “Network communications parameters” on page 157.

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.

**Examples**

- The following connection string fragment starts the TCP/IP protocol only:

```
CommLinks=tcpip
```

- The following connection string fragment starts the shared memory protocol and searches for the database server over shared memory. If the search fails, it then starts the TCP/IP port searching for the host kangaroo in addition to servers on the immediate TCP/IP network.

```
CommLinks=tcpip (HOST=kangaroo) , shmem
```

## **ConnectionName connection parameter [CON]**

**Function**

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]**

**Function**

For use when starting a database that is not already running. The DatabaseFile connection parameter indicates which database file you want to load and connect to.

If you want 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. For details, see “Reconnecting after you restore” in *Sybase IQ System Administration Guide*.

Usage	Embedded databases
Values	<i>String</i>
Default	There is no default setting.
See also	Chapter 1, “Running the Database Server,” <i>Sybase IQ Utility Guide</i>
Description	<p>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.</p> <ul style="list-style-type: none"><li>• 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.</li><li>• If the filename does not include an extension, a file of name .db is looked for.</li><li>• 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.</li><li>• 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.</li></ul>

You can also use UNC filenames and Novell NetWare Directory Services file names.

---

### **Caution**

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.

---

See also	“DatabaseName connection parameter [DBN]”
Example	To load and connect to the sample database (installed in directory <i>C:\Program Files\Sybase\ASIQ-12_7\demo</i> on Windows), use the following DBF parameter:

```
DBF=C:\Program Files\Sybase\ASIQ-12_7\demo\asiqdemo.db
```

## DatabaseName connection parameter [DBN]

Function	For use when connecting to a database that is already running. Identifies a loaded database to which a connection needs to be made.
	If you want to connect to a database that is not already running, use the DatabaseFile (DBF) parameter.
Usage	Running network servers
Values	<i>String</i>
Default	There is no default setting.
Description	<p>Whenever 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 filename with the extension and path removed.</p> <p>If the database you want to connect to is already running, you should specify the database name rather than the database file.</p> <p>A connection will only occur if the name of the running database matches the name that is specified in the DatabaseName (DBN) parameter.</p>

---

**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.

---

See also	For HTTP connections, see “DatabaseName communication parameter [DBN]” on page 161.
Examples	<ul style="list-style-type: none"><li>To start a database file named <i>cities.db</i> and rename the database Kitchener, you can use the following command:<pre>start_asiq cities.db -n Kitchener</pre></li><li>Assuming you have run the above command, you can successfully connect to the running database named Kitchener as follows:<pre>DBN=Kitchener</pre></li><li>Alternatively, you could use the following to successfully connect to the running database named Kitchener:<pre>DBN=Kitchener;DBF=cities.db</pre></li><li>However, specifying the following would fail to connect to the database named Kitchener:<pre>DBF=cities.db</pre></li></ul>

## **DatabaseSwitches connection parameter [DBS]**

Function	To provide database-specific switches when starting a database.
Usage	Connecting to a server when the database is not loaded.
Default	No switches
See also	<i>Chapter 1, “Running the Database Server,” Sybase IQ Utility Guide</i> “StartLine connection parameter [START]”
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 /ASIQ-12_7/demo/asiqdemo.db (DBF parameter), names it as my_db (DBS parameter) and connects to the database of that name (DBN parameter).  <pre>dbcollat -c "uid=DBA;pwd=SQL;dbf=/ASIQ-12_7/demo/asiqdemo.db; dbn=my_db; dbs=-n my_db" /tmp/temp.col</pre>

## **DataSourceName connection parameter [DSN]**

Function	Tells the ODBC driver manager or Embedded SQL library where to look in the .odbci 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.
See also	“FileDataSourceName connection parameter [FileDSN]” on page 149
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

## DBKEY connection parameter [DBKEY]

Function	To start 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.
See also	Starting the database server in <i>Sybase IQ Utility Guide</i> .
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 <i>marvin.db</i> that is already running.  ... UID=dba;PWD=sql;DBF=marvin.db;DBKEY='is!seCret'  ...

## DisableMultiRowFetch connection parameter [DMRF]

Function	To turn 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	<ul style="list-style-type: none"><li>The following connection string fragment prevents prefetching:</li></ul>

DMRF=Yes

## EngineName connection parameter [ENG]

Function	Synonym for ServerName. The name of a running database server to which you want to connect.
Usage	Network servers
Values	<i>String</i>
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.

Protocol	Truncation Length
UNIX shared memory	31 bytes
non-UNIX shared memory	40 bytes
TCP/IP	40 bytes
Named Pipes	8 bytes

See also	“Identifiers” in Chapter 3, “SQL Language Elements,” in <i>Sybase IQ Reference Manual</i> . “Database switches” in Chapter 1, “Running the Database Server,” <i>Sybase IQ Utility Guide</i> “How connection parameters work” on page 74
Examples	Connect to a server named Guelph:

ENG=Guelph

## EncryptedPassword connection parameter [ENP]

Function	To provide 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	<i>String</i>
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	“How connection parameters work” on page 74

## Encryption connection parameter [ENC]

Function	To encrypt 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	<i>String</i>
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: <ul style="list-style-type: none"><li>• <b>none</b> accepts communication packets that are not encrypted. This value is equivalent to NO in previous versions of Sybase IQ.</li><li>• <b>simple</b> 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.</li><li>• <b>ECC_TLS</b> (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:<ul style="list-style-type: none"><li>• <b>trusted_certificates</b> specify the certificate file the client uses to authenticate the server.</li></ul></li></ul>

- **certificate\_company** specify the value for the organization field. The server's value and the client's value must match.
- **certificate\_unit** specify the value for the organization unit field. The server's value and the client's value must match.
- **certificate\_name** specify the certificate's common name. The server's value and the client's value must match.
- **RSA\_TLS** accepts communication packets that are encrypted using RSA encryption technology. To use this type of encryption, both the server and the client must be operating on Solaris, Linux, and all supported Windows operating systems, and the connection must be over the TCP/IP port. UNIX platforms, except for Solaris and Linux, do not recognize the client or server RSA\_TLS parameter. To authenticate the server, the Certicom software verifies that the server's certificate values match any values you supply about the client using the following arguments:
  - **trusted\_certificates** specify the certificate file the client uses to authenticate the server.
  - **certificate\_company** specify the value for the organization field. The server's value and the client's value must match.
  - **certificate\_unit** specify the value for the organization unit field. The server's value and the client's value must match.
  - **certificate\_name** 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 server command-line switch in “Starting the database server” in *Sybase IQ Utility Guide*

“`CONNECTION_PROPERTY` function [System]” in *Sybase IQ Reference Manual*

- 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]

Function	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	<i>String</i>
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: <pre>[Sample File Data Source] ENG=asicdemo DBA=DBA PWD=SQL</pre>
See also	“DataSourceName connection parameter [DSN]” on page 144

## Idle connection parameter [IDLE]

Function	To specify the connection's idle timeout period.
----------	--

Usage	Anywhere except with TDS and Shared Memory connections. Shared Memory and TDS connections (including jConnect) ignore the Sybase IQ Idle (IDLE) connection parameter.
Values	<i>Integer</i>
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_asiq. 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 Chapter 1, “Running the Database Server” in <i>Sybase IQ Utility Guide</i>
Example	<ul style="list-style-type: none"><li>The following connection string fragment sets the timeout value for this connection to 10 minutes:  <code>"ENG=myeng;LINKS=tcpip;IDLE=10"</code></li></ul>

## Integrated connection parameter [INT]

Function	To use the integrated login facility.
Usage	Anywhere
Values	YES, NO
Default	NO
See also	“LOGIN_MODE option” in <i>Sybase IQ Reference Manual</i>
Description	The Integrated parameter has the following settings: <ul style="list-style-type: none"><li><b>Yes</b> An integrated login is attempted. If the connection attempt fails and the LOGIN_MODE option is set to Mixed, a standard login is attempted.</li><li><b>No</b> This is the default setting. No integrated login is attempted.</li></ul>

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]

Function	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 ASLANG environment variable or the installer.
Description	<p>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.</p> <p>If no language is specified, the default language is used. The default language is the language specified by, in order, the ASLANG environment variable or the installer.</p> <p>For a list of language codes, see “Understanding the locale language” in <i>Sybase IQ System Administration Guide</i>.</p> <p>This connection parameter only affects the connection. Messages returned from Adaptive Server Anywhere's various tools and utilities appear in the default language, while the messages returned from the server appear in the connection's language.</p>
See also	“How connection parameters work” in <i>Sybase IQ System Administration Guide</i>

## LazyClose connection parameter [LCLOSE]

Function	Enabling this option causes the CLOSE <i>cursor-name</i> 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	<p>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 <i>cursor-name</i> database request is queued.</p> <p>Enabling this option can improve performance if:</p> <ul style="list-style-type: none"><li>• Your network exhibits poor latency</li><li>• Your application sends many cursor open and close requests</li></ul> <p>Note that in rare circumstances, canceling the next request after the CLOSE <i>cursor-name</i> 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.</p>
See also	<p>“How connection parameters work” on page 74</p>

## **LivenessTimeout connection parameter [LTO]**

Function	To control the termination of connections when they are no longer intact.
Usage	Network server on TCP/IP communications protocols.
Values	All platforms except non-threaded UNIX applications.
Default	<i>Integer</i> (in seconds)
	120
Description	<p>If no LivenessTimeout value is set, the liveness timeout is controlled by the setting on the server, which defaults to 120 seconds.</p> <p>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.</p> <p>Liveness packets are sent when a connection has not sent any packets for between one third and two thirds of the LivenessTimeout value.</p> <p>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.</p>

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]

Function	To send client error messages and debugging messages to a file.
Usage	Anywhere
Values	<i>String</i>
See also	“How connection parameters work” on page 74
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 <i>error.log</i> in the current working directory for the client:

```
...
LogFile=error.log
...
```

## Password connection parameter [PWD]

Function	To provide a password for the connection.
Usage	Anywhere

Default	No password provided
See also	<a href="#">“EncryptedPassword connection parameter [ENP]” on page 146</a>
Description	<p>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.</p> <p>The password parameter is not encrypted. If you are storing passwords in a connection profile, you should use the <code>EncryptedPassword</code> parameter. Sybase Central and the Sybase IQ ODBC configuration tool both use encrypted parameters.</p> <p>If both <code>Password</code> and <code>EncryptedPassword</code> are specified, <code>Password</code> takes precedence.</p>
Examples	<p>The following connection string fragment supplies the user ID <code>DBA</code> and password <code>SQL</code>.</p> <pre>uid=DBA;pwd=SQL</pre> <p>Alternatively, you can set these parameters in the User ID and Password text boxes in the connection window.</p>

## PrefetchBuffer connection parameter [PBUF]

Function	Set the maximum amount of memory for buffering rows, in kilobytes.
Usage	Anywhere
Values	<i>Integer</i>
Default	64 (KB)
See also	<a href="#">“PrefetchRows connection parameter [PROWS]”</a>
Description	<p>The PrefetchBuffer connection parameter controls the memory allocated on the client to store prefetched rows. 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 <code>PrefetchRows</code> and <code>PrefetchBuffer</code> connection parameters.</p> <p>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.</p>

Example	The following connection string fragment could be used to determine if the PrefetchBuffer memory limit is reducing the number of rows prefetched:
---------	---

```
...prefetchrows=100;logfile=c:\ client.txt
```

The following string could be used to increase the memory limit to 256K:

```
...prefetchrows=100;prefetchbuffer=256
```

## PrefetchRows connection parameter [PROWS]

Function	Set the maximum number of rows to prefetch when querying the database.
Usage	Anywhere
Default	10
See also	“PrefetchBuffer connection parameter [PBUF]”
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.

Example	The following connection string fragment sets the number of prefetched rows to 100:
---------	---

```
...prefetchrows=100;...
```

## ServerName connection parameter [ENG]

Synonym for “EngineName connection parameter [ENG]”.

For more information, see “EngineName connection parameter [ENG]” on page 146

## **StartLine connection parameter [START]**

Function	To start 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.  For a detailed description of available command line switches, see Chapter 1, “Running the Database Server,” <i>Sybase IQ Utility Guide</i> .
Examples	<ul style="list-style-type: none"><li>The following data source fragment starts a database server with a cache of 32MB.</li></ul>

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

## **Unconditional connection parameter [UNC]**

Function	To stop a server using <i>dbstop</i> even when there are connections to the server.
Usage	Anywhere
Default	No
See also	“The Stop utility ( <i>dbstop</i> )” in the <i>Sybase IQ Utility Guide</i> .
Description	The <i>dbstop</i> 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	<ul style="list-style-type: none"><li>The following command line shuts down the server unconditionally:</li></ul>

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

## **Userid connection parameter [UID]**

Function	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_asiq -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_asiq -x "tcpip(PARM1=value 1;PARM2=value  
2;...),..."  
start_asiq -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]	

## Broadcast communication parameter [BCAST]

Usage	TCP/IP
Values	<i>String</i> (in the form of an IP address)
Default	Broadcasts to all addresses on the same subnet
Description	BROADCAST specifies the IP address used by your TCP/IP protocol implementation to identify a broadcast message.  Broadcast addresses consist of the network IP address portion, with 255 as the remaining integers. For example:
	<ul style="list-style-type: none"><li>• If the network portion is 95 (class A), then the broadcast address would be 95.255.255.255</li><li>• If the network portion is 132.10 (class B), then the broadcast address would be 132.10.255.255</li></ul>

- If the network portion is 197.31.175 (class C), then the broadcast address would be 197.31.175.255

The broadcast IP address 255.255.255.255 broadcasts to all subnets.

## BroadcastListener communication parameter [BLISTENER]

Usage	TCP/IP, Server side
Values	YES, NO
Default	YES
Description	This option allows you to turn broadcast listening OFF for this port. Using -sb 0 is the same as specifying BroadcastListener=NO on TCP/IP.
See also	“Starting the database server” <i>Sybase IQ Utility Guide</i> .
Example	<ul style="list-style-type: none"><li>Start a server that accepts TCP/IP connections that use BroadcastListener=NO:  <code>start_asiq -x tcpip (BroadcastListener=NO)</code></li></ul>

## Certificate communication parameter

Usage	HTTP, HTTPS
Values	<i>String</i>
Default	There is no default certificate name.
Description	This option allows you to specify the name of an encryption certificate. The password for this certificate must be specified with the Certificate_Password parameter.
Example	<ul style="list-style-type: none"><li>Start a server that requires web connections to use a particular encryption certificate.  <code>start_asiq -xs http (Certificate=cert.file;Certificate_Password=sec ret) ...</code></li></ul>
See also	“Host parameter (IP)” on page 163 “DoBroadcast parameter [DBROAD]” on page 161 “ServerPort parameter (PORT)” on page 170

## Certificate\_Password communication parameter

Usage	HTTP, HTTPS
Values	<i>String</i>
Default	There is no default certificate password.
Description	This option allows you to specify the password that matches the encryption certificate specified by the Certificate parameter.
Example	<ul style="list-style-type: none"><li>Start a server that requires web connections to use a particular encryption certificate.<pre>start_asiq -xs http(Certificate=cert.file;Certificate_Password=sec ret) ...</pre></li></ul>

## ClientPort parameter [CPort]

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. It is recommended that you do not use this parameter unless you need to for firewall reasons. <p>The ClientPort option 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.</p> <p>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.</p>
Examples	<ul style="list-style-type: none"><li>The following string make a connection from an application using port 6000 to a server named my_server using port 5000:</li></ul>

```
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_server
```

## DatabaseName communication parameter [DBN]

Usage	HTTP, HTTPS
Values	AUTO, REQUIRED, <i>database-name</i>
Default	AUTO
Description	<p>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.</p> <p>If this parameter is set to REQUIRED, the URI must specify a database name.</p> <p>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.</p> <p>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.</p>
Example	<ul style="list-style-type: none"><li>• The following command starts two databases, but permits only one of them to be accessed via HTTP.</li></ul> <pre>start_asiq -xs http(dbn=web) asiqdemo.db web.db</pre>

## DoBroadcast parameter [DBROAD]

Usage	TCP/IP (all platforms)
Values	ALL, NONE, DIRECT (Client side)
	YES, NO (Server side)

Default	ALL
Description	<p><b>Client usage</b> With <code>DoBroadcast=ALL</code> (formerly <code>DoBroadcast=YES</code>) a broadcast is performed to search for a server. The broadcast goes first to the local subnet. If <code>HOST=</code> is specified, broadcast packets are also sent to each of the hosts. For TCP, all broadcast packets are UDP packets.</p> <p>With <code>DoBroadcast=DIRECT</code> (formerly <code>DoBroadcast=NO</code>), 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 <code>HOST (IP)</code> communication parameter. If you specify <code>DoBroadcast=DIRECT</code>, the <code>HOST (IP)</code> communication parameter is required.</p> <p>Specifying <code>DoBroadcast=NONE</code> causes no UDP broadcasts to be used. A TCP/IP connection is made directly with the <code>HOST/PORT</code> specified, and the server name is verified. With TCP/IP, you can choose not to verify the server name by setting the <code>VerifyServerName (VERIFY)</code> communication parameter to <code>NO</code>. The <code>HOST (IP)</code> communication parameter is a required parameter, while the <code>ServerPort (PORT)</code> communication parameter is optional.</p> <p>For <code>DIRECT</code> and <code>NONE</code>, you must specify the server host with the <code>HOST</code> option.</p> <p><b>Server usage</b> Setting <code>DoBroadcast=NO</code> 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.</p>
Example	<ul style="list-style-type: none"><li>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 <code>silver</code>.<pre>dbisql -x tcpip (DOBROADCAST=NO;HOST=silver) asiqdemo</pre></li><li>On UNIX, the options must be enclosed in quotation marks:<pre>dbisql -x "tcpip (DOBROADCAST=NO;HOST=silver)" asiqdemo</pre></li></ul>

## DLL parameter

Usage	TCP/IP (Windows)
Description	To support untested TCP/IP protocol stacks where the required networking interface functions are in DLLs that differ from the default protocol stack. The client or server looks for its required functionality in the named DLLs.

---

Values	<i>String</i>
Default	On all supported Windows platforms, the default is <i>ws2_32.dll</i> (Winsock 2.0).
Example	The following command starts a server with the protocol interface functions in <i>abc.dll</i> :

```
asiqsr12 -x tcpip(dll=abc.dll) asiqdemo
```

## Host parameter (IP)

Usage	TCP/IP (all platforms) Server and client sides
See also	“ClientPort parameter [CPort]” on page 160
Where	Server and client sides
Description	<p>HOST specifies additional machines outside the immediate network to be searched by the client library. On the server, the search is carried out to avoid starting a server with a duplicate name.</p> <p>For TCP/IP, the <i>hostname</i> or a dot-separated IP address may be used.</p> <p>The server prints this addressing information during startup if the <i>-z</i> switch is used. In addition, the application writes this information to its logfile if LogFile is specified (Debug is set to TRUE).</p> <p>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 <i>Host=nnn.nn.nnn.nnn;ServerPort=pppp</i>.</p> <p>IP and HOST are synonyms when using TCP/IP.</p>

Values	<i>String</i>
Default	No additional machines.
Example	<ul style="list-style-type: none"> <li>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 <i>asiqdemo</i>:</li> </ul>

```
...ENG=asiqdemo 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)" asiqdemo
```

- 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.

```
LINKS=tcpip (HOST=my_server,kangaroo;PORT=2639)
```

## LDAP communication parameter [LDAP]

Usage	TCP/IP (Server side only)
Values	YES, NO, or <i>filename</i>
Default	ON
Description	<p>The default <i>filename</i> is <i>asldap.ini</i></p> <p>Having the database server register itself with an LDAP server allows clients (and the Locate utility [<code>dblocate</code>]) to query the LDAP server. This allows clients running over a WAN or through a firewall to find servers without specifying the IP address. It also allows the Locate utility (<code>dblocate</code>) to find such servers.</p> <p>Specifying <code>LDAP=filename</code> turns LDAP support on and uses the specified file as the configuration file. Specifying <code>LDAP=YES</code> turns LDAP support on and uses <i>asldap.ini</i> as the configuration file.</p> <p>LDAP is only used with TCP/IP, and only on network servers.</p>
See also	For more information, see “Connecting using an LDAP server” in <i>Adaptive Server Anywhere Database Administration Guide</i> .

## LocalOnly communication parameter [LOCAL]

Usage	TCP/IP, HTTP, HTTPS
Values	YES, NO
Default	NO
Description	The LocalOnly (LOCAL) communication parameter allows a client to choose to connect only to a server on the local machine, if one exists. 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 <code>DoBroadcast=ALL</code> (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 158

## LogFile communication parameter [LOG]

Usage	HTTP, HTTPS
Values	<i>Filename</i>
Default	None
Description	Specify the name of the file to which the database server is to write information about web requests.
See also	<a href="#">“LogFormat communication parameter [LF]” on page 165</a> <a href="#">“LogMaxSize communication parameter [LSIZE]” on page 166</a> <a href="#">“LogOptions communication parameter [LOPT]” on page 167</a>

## LogFormat communication parameter [LF]

Usage	HTTP, HTTPS
Values	<i>Format-string</i>
Default	<code>@T - @W - @I - @P - "@M @U @V" - @R - @L - @E</code>
Description	This parameter controls the format of messages written to the log file and which fields appear in them. If they appear in the string, the current values are substituted for the following codes as each message is written. <ul style="list-style-type: none"><li>• <b>@@</b> The @ character.</li><li>• <b>@B</b> Date and time that processing of the request started, unless the request could not be queued due to an error.</li><li>• <b>@C</b> Date and time that the client connected.</li></ul>

- **@D** Name of the database associated with the request.
- **@E** Text of the error message, if an error occurred.
- **@F** Date and time that processing of the request finished.
- **@I** IP address of the client.
- **@L** Length of the response, in bytes, including headers and body.
- **@M** HTTP request method.
- **@P** Listener port associated with the request.
- **@Q** Date and time that the request was queued for processing, unless the request could not be queued due to an error.
- **@R** Status code and description of the HTTP response.
- **@S** HTTP status code.
- **@T** Date and time that the current log entry was written.
- **@U** Requested URI.
- **@V** Requested HTTP version.
- **@W** Time taken to process the request (@F – @B), or 0.000 if the request was not processed due to an error.

See also

“LogFile communication parameter [LOG]” on page 165

“LogMaxSize communication parameter [LSIZE]” on page 166

“LogOptions communication parameter [LOPT]” on page 167

## LogMaxSize communication parameter [LSIZE]

Usage	HTTP, HTTPS
Values	<i>Size</i>
Default	0
Description	When the log file reaches the stated size, it is renamed and another log file is created. If LogMaxSize is zero, the log file size is unlimited.
See also	“LogFile communication parameter [LOG]” on page 165 “LogFormat communication parameter [LF]” on page 165 “LogOptions communication parameter [LOPT]” on page 167

## LogOptions communication parameter [LOPT]

Usage	HTTP, HTTPS
Values	NONE, OK, INFO, ERRORS, ALL, <i>status-codes</i> , REQHDRS, RESHDRS, HEADERS
Default	ALL
Description	<p>The values available include keywords that select particular types of messages, and HTTP status codes. Multiple values may be specified, separated by commas.</p> <p>The following keywords control which categories of messages are logged:</p> <ul style="list-style-type: none"><li>• <b>NONE</b> Log nothing.</li><li>• <b>OK</b> Log requests that complete successfully (20x HTTP status codes).</li><li>• <b>INFO</b> Log requests that return over or not modified status codes (30x HTTP status codes).</li><li>• <b>ERRORS</b> Log all errors (40x and 50x HTTP status codes)</li><li>• <b>ALL</b> Log all requests.</li></ul> <p>The following common HTTP status codes are also available. They can be used to log requests that return particular status codes:</p> <ul style="list-style-type: none"><li>• <b>C200</b> OK</li><li>• <b>C400</b> Bad request</li><li>• <b>C401</b> Unauthorized</li><li>• <b>C403</b> Forbidden</li><li>• <b>C404</b> Not found</li><li>• <b>C408</b> Request timeout</li><li>• <b>C501</b> Not implemented</li><li>• <b>C505</b> Service unavailable</li></ul> <p>In addition, the following keywords may be used to obtain more information about the logged messages:</p> <ul style="list-style-type: none"><li>• <b>REQHDRS</b> When logging requests, also write request headers to the log file.</li><li>• <b>RESPHDRS</b> When logging requests, also write response headers to the log file.</li></ul>

- **HEADERS** 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 165

“LogFormat communication parameter [LF]” on page 165

“LogMaxSize communication parameter [LSIZE]” on page 166

## MaxConnections communication parameter [MAXCONN]

Usage	HTTP, HTTPS
Values	<i>Size</i>
Default	Number of licensed connections
Description	The number of simultaneous connections accepted by the server. The value 0 indicates no limit.
See also	“MaxRequestSize communication parameter [MAXSIZE]” on page 168

## MaxRequestSize communication parameter [MAXSIZE]

Usage	HTTP, HTTPS
Values	<i>Size</i>
Default	100KB
Description	The size of the largest request accepted by the server. If the size of a request exceeds this limit, the connection is closed and a 413 ENTITY TOO LARGE response is returned to the client. This value limits only the size of the request, not that of the response. The value 0 disables this limit, but should be used with extreme caution. Without this limit, a rogue client could overload the server or cause it to run out of memory.
See also	“MaxConnections communication parameter [MAXCONN]” on page 168

## MyIP parameter [ME]

Usage	TCP/IP, HTTP, HTTPS
Values	<i>String</i>

Description	<p>The MyIP parameter is provided for machines with more than one network adapter.</p> <p>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.</p> <p>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.</p> <p>On Windows platforms, this option can be used multiple times for machines with multiple IP addresses.</p> <p>Separate multiple IP addresses with commas. You can optionally append a port number to the IP address, separated by a colon.</p>
Example	<ul style="list-style-type: none"> <li>The following Windows command line (entered all on one line) instructs the server to use two network cards, one with a specified port number.           <pre>asiqsrv12 -x tcpip(MyIP=192.75.209.12:2367,192.75.209.32) c:\sybase\ASIQ-12_7\demo\asiqdemo.db</pre> </li> <li>The following connection string fragment instructs the client to make no attempt to determine addressing information.           <pre>...CommLinks= tcpip(MyIP=NONE) ...</pre> </li> </ul>

## PreFetchOnOpen communication parameter

Usage	ODBC
Values	YES, NO
Default	NO
Description	<p>Enabling this option sends a prefetch request with a cursor open request, thereby eliminating a network request to fetch rows each time a cursor is opened. 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.</p> <p>Calling ODBC's SQLExecute or SQLExecDirect on a query or stored procedure which returns a result set causes a cursor open.</p>

Enabling this option can improve performance if your:

- network exhibits poor latency
- application sends many cursor open and close requests

## **ReceiveBufferSize communication parameter [RCVBUFSZ]**

Usage	TCP/IP
Values	<i>Integer</i>
Default	Machine-dependent
Description	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.

## **SendBufferSize communication parameter [SNDBUFSZ]**

Usage	TCP/IP
Values	<i>Integer</i>
Default	Machine-dependent
Description	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.

## **ServerPort parameter (PORT)**

Usage	TCP/IP (all platforms), HTTP, HTTPS
Values	<i>Integer</i>
Default	The default value for TCP/IP is 2638. The default value for HTTP is 80. The default value for HTTPS is 443.
Description	<p>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.</p> <p>In the case of the database server, the ServerPort option designates the port number on which to communicate using TCP/IP.</p>

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. 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_asiq -x tcpip  
c:\sybase\ASIQ-12_7\demo\asiqdemo.db
```

Port number 2638 is now taken.

- 2 Attempt to start another database server:

```
start_asiq -x tcpip  
c:\sybase\ASIQ-12_7\demo\asiqdemo2.db
```

This fails with an error “Unable to initialize communication links,” 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_asiq -x tcpip(ServerPort=2639)  
c:\sybase\ASIQ-12_7\demo\asiqdemo2.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_asiq -xs http(port=8080) -n server3 web.db
```

## Sessions parameter

**Usage**

NetBIOS. Server side only.

Description	Sets the maximum number of clients that can communicate with the server at one time through a single LAN adapter. 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 <i>commands</i> 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.

The following statement starts a server with a database named asiqdemo, allowing 200 NetBIOS connections.

```
asiqsvr12 -x netbios(sessions=200) asiqdemo.db
```

## TDS parameter

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	<ul style="list-style-type: none"><li>The following command starts a database server using the TCP/IP protocol, but disallowing connections from Open Client or jConnect applications.</li></ul>

```
start_asiq -x tcpip(TDS=NO) ...
```

## Timeout parameter [TO]

Usage	TCP/IP (all platforms), HTTP, HTTPS
Values	<i>Integer</i> , in seconds
Default	5
Description	TIMEOUT specifies the length of time, in seconds, to wait for a response when establishing communications and when disconnecting. 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 parameter [Verify]

Usage	TCP/IP (Client side only)
Values	YES, NO
Default	YES
Description	<p>Specifies whether the server name is verified when connecting to this host. Normally you should not set this option. It is used only for connecting to multiplex query servers when you need to balance query loads among these servers.</p> <p>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.</p> <p>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.</p> <p>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.</p>
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 query 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 query server based on the workload of the machine.

# Working with Database Objects

## About this chapter

This chapter describes the mechanics of creating, altering, and deleting databases and database objects such as tables, views, and indexes. Procedures, which are also database objects, are discussed in Chapter 8, “Using Procedures and Batches.”

---

**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 Adaptive Server Anywhere tables. If you want to create tables in the Catalog Store, you need to refer to the Adaptive Server Anywhere documentation.

---

## Contents

Topic	Page
Building your Sybase IQ databases	176
Working with database objects	183
Working with dbspaces	216
Dbspace management example	228
Working with tables	239
Working with views	249
Working with indexes	254

## Building your Sybase IQ databases

This section introduces you to the steps and tools used to create a database. It also explains decisions you need to make about where to store the data, how much space it will require, and who will be able to define or modify database objects.

### Designing your database

It's important to design your database before you actually create it. The right database design can make a major difference in the usefulness of your data, and the speed with which you can retrieve it.

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.

An 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 using the procedures in this chapter to create 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 two utilities for working with database objects: Sybase Central and DBISQL. In addition, Warehouse Architect can be used for designing and creating whole data warehouses.

## Using Sybase Central to work with database objects

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 query servers, tables, procedures, views, indexes, users and groups.

If you use the Sybase IQ multiplex feature, Sybase recommends that you use Sybase Central to manage database objects. In a multiplex, Sybase Central enables easy creation query servers and dbspaces, and also lets you start and shut down servers. For more information about the multiplex feature, see “Multiplex capability” on page 8. DML and DDL statements are the same for multiplex and non-multiplex Sybase IQ databases, except for a few restrictions described in “Updating multiplex databases” on page 212 and “Multiplex management without Sybase Central” on page 201.

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 Chapter 4, “Managing Databases” in *Introduction to Sybase IQ*.

## Using DBISQL to work with database objects

Interactive SQL (DBISQL) is a utility for entering SQL statements. If you are using DBISQL 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. For more information, see “How locking works” on page 485.

---

If you use a tool other than DBISQL, all the information in this chapter concerning SQL statements still applies.

**DBISQL command file** A DBISQL 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 ( . . . );
. . .
```

A DBISQL command file usually carries the extension *.sql*. To execute a command file, either paste the contents of the file into the DBISQL 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 command:

```
read makedb
```

reads the DBISQL commands in the file *makedb.sql*.

For more information about reading a file into the command window, see **READ statement [DBISQL]** in the *Sybase IQ Reference Manual*.

## **A step-by-step overview of database setup**

Creating an IQ database is part of a larger setup process that begins with installation and ends when your database is available to users. This section summarizes the steps in setting up an IQ database and the objects in it.

### ❖ **Setting up an IQ database**

- 1 Install and configure Sybase IQ.

This step installs the client and server environment and the *asiqdemo* database. See the *Sybase IQ 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 either Sybase Central or the CREATE DATABASE statement.

For Sybase Central (the easiest way to use multiplex capability), see “Using Sybase Central to create an IQ database” on page 186.

For CREATE DATABASE instructions, see “Creating a database with SQL” on page 183.

- 3 Create the tables in your IQ database.

Use the CREATE TABLE statement or the Sybase Central table editor. See “Working with tables” on page 239

- 4 Create indexes for the tables.

Use the CREATE INDEX statement or the Sybase Central Index Wizard. You can also create certain indexes automatically when you create your tables. See Chapter 6, “Using Sybase IQ Indexes.”

- 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. See Chapter 7, “Moving Data In and Out of Databases.”

## 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”.

## Creating a dummy table 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. For details on the performance monitor, see Chapter 6, “Monitoring and Tuning Performance” in *Sybase IQ Performance and Tuning Guide*.

## Extending data definition privileges

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.

The DBA and object owners can grant authority to individual users and to groups of users. For complete information, see Chapter 12, “Managing User IDs and Permissions.” You can also use the -gu command-line option to set the permission level required to create or delete a database.

## Selecting a device type

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.

---

## Allocating space for databases

All Sybase IQ databases are preallocated, whether they reside in a file system or a raw partition.

Each database includes multiple **dbspaces**. A dbspace is a logical name for a database file. The Catalog Store, the IQ Store, and the Temporary Store all consist of dbspaces. 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 the data among all existing dbspaces in that store with any room available. You cannot control how data is distributed among the dbspaces within a store. 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.

If possible, do not allocate all of your disk space to your IQ database. Keep ten percent in reserve. Sybase IQ needs this space to handle out-of-space conditions gracefully.

**When to create dbspaces**

When possible, create all dbspaces when you create the database, rather than adding them gradually as old ones become full. This approach ensures that your dbspaces will be filled more evenly, and thus helps improve disk I/O.

**Space requirements for IQ 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, Sybase IQ prompts you to create another dbspace, and then continues the operation after you add the dbspace.

**Space requirements for Temporary Stores**

In addition to any temporary tables you define explicitly, 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.

**Creating development databases**

It is generally a good idea to create separate databases for debugging purposes. Because development work can increase the possibility of a server failure, it is best to avoid it on production databases.

## **Estimating space and dbspaces required**

To avoid difficulties when a database or a particular dbspace is full, you should estimate your 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 the *Sybase IQ Reference Manual* for syntax and usage notes for each procedure.

Running the procedures in the sequence that follows can help you avoid running out of space for your objects.

- 1 Run the stored procedure `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. Run the procedure once for each table that you plan to create, as follows:

```
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 Total 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, doing a separate calculation for minimum and maximum join space.
- 6 Run the stored procedure sp\_iqestdbspaces to determine how many dbspaces to create from the given space and what size they should be. Use the minimum and maximum total index sizes calculated in step number 5 as the *minsize* and *maxsize* parameters for this procedure, as follows:

```
sp_iqestdbspaces (dbsize [,iqpagesize]  
[,minsize] [,maxsize] ...)
```

All these calculations are estimates. Results vary based on the columns and indexes you create for your database. For more information on these stored procedures, see the *Sybase IQ Reference Manual*.

## Working with database objects

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.

This chapter describes the SQL statements for defining database objects. You can use these statements directly if you are building your database from an interactive SQL tool, such as DBISQL. Even if you are using 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.

For more advanced use, database design tools such as Sybase PowerDesigner provide a more thorough and reliable approach to developing well-designed databases.

## Creating a database with SQL

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	<i>dbname.iq</i>
IQ_SYSTEM_MSG	Message log file	<i>dbname.msg</i>
IQ_SYSTEM_TEMP	Temporary IQ Store file	<i>dbname.iqtmp</i>
SYSTEM	Catalog Store file	<i>dbname.db</i>

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 Temporary IQ 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 ASTMP environment variable on UNIX.

---

Once the database is created, you can connect to it and build the tables and other objects that you need in the database.

Before you create  
your database

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. 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:

- Start the database server without a database by specifying only `-n enginename` on the startup command.
- 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.)

- In Sybase Central, in the Create Database Wizard, choose Utility Server as the engine that will create the database.

For more information on the utility database and its security, see “Utility database server security” on page 564.

If you are creating an IQ database for the first time, see the *Introduction to Sybase IQ* for assistance.

---

**Note** If the server is started with the `-m` server option, you cannot create a database.

---

#### Locating and moving database files

When you create a database, you specify its location. Before you do so, consider whether you will ever need to move the database.

The IQ Catalog (`.db`) and transaction log (`.log`) files can be safely moved. *Never attempt to copy a running database.* If you use relative pathnames 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) pathnames 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. See Chapter 14, “Data Backup, Recovery, and Archiving” for more information.

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. This leads to poor I/O performance, can overload the local area network, and can lead to problems in the dbspace file. On UNIX platforms, avoid Network File System (NFS) mounted file systems. On Windows platforms, do not place dbspace files on Network Drives owned by another node.

---

If your IQ requirements are large and complex enough that you need multiple physical systems, consider using Sybase IQ multiplex functionality. See “Multiplex capability” on page 8 for an overview.

#### Database file compatibility

Sybase IQ servers cannot manage databases created with versions prior to Sybase IQ 12.5; likewise, old servers cannot manage new databases.

Using Sybase Central  
to create an IQ  
database

To create an IQ database in Sybase Central, select the Sybase IQ plug-in, click the Utilities tab, then double-click Create Database. The Create Database Wizard leads you through the process. For a multiplex database, see “Creating databases with multiplex functionality” on page 193. For a nonmultiplex database, see “Creating databases” in Chapter 4, “Managing Databases,” in *Introduction to Sybase IQ*.

Using the CREATE  
DATABASE statement

You can use the CREATE DATABASE statement to create IQ databases. You must specify the filename 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’. Note that the default login is now user ID DBA and password SQL (uppercase). By default, passwords are case sensitive for a case-sensitive database, and case-insensitive for a case-insensitive database. 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. See also “Using relative pathnames.”
- 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.

---

**Note** Unless CREATE DATABASE commands include CASE IGNORE or PASSWORD CASE IGNORE, connections to newly created databases must be made with case-sensitive user ID/password combinations.

---

For a full description of all parameters, see CREATE DATABASE statement in the *Sybase IQ Reference Manual*. Following are several examples of creating an IQ database.

#### Using relative pathnames

You can create a database using a relative or fully qualified pathname for each of the files for the database. Sybase recommends that you create databases with relative pathnames. If you specify absolute pathnames, you will not be able to move files to a different pathname without backing up and restoring the database.

If your database is on UNIX, you can define a symbolic link for each pathname, as described in the *Sybase IQ Reference Manual*.

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 pathname. Using a configuration file to start the server ensures that you start the server from a consistent location.

---

#### Specifying an IQ PATH

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 shutdown the database and you may have to perform a backup and restore.

You can add space on a different drive, as described in “Adding dbspaces” but you can only use this additional space for new data. You cannot readily move a particular table, index, or rows of data from one location to another. You would need to drop the table or index, recreate it, and reload it; or you would need to delete those rows, and reinsert them.

Each operating system has its own format for raw device names. See Chapter 8, “Physical Limitations” in the *Sybase IQ Reference Manual* for an important note about initializing raw devices on Sun Solaris.

**Table 5-1: Raw device names on UNIX**

UNIX Platform	Example
AIX	/dev/rraw121v
HP-UX	/dev/vg03/rrchee12g
Sun Solaris	/dev/rsd0c

**Table 5-2: 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. (This behavior results when the ESCAPE\_CHARACTER option is set ON for the database.)

You must *always* double the backslash when naming raw devices on Windows in SQL statements. See Example 4.

#### 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:'
```

## Choosing an IQ page size

You set a page size for the IQ Store with the IQ PAGE SIZE option. This option determines memory and disk use. 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 Chapter 5, “Managing System Resources,” in *Sybase IQ Performance and Tuning Guide*.

#### Setting IQ 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.

## Specifying the size of your database

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 partition size.
- For operating system files you can rely on the defaults listed below; or specify a value based on the size of your data, from the required minimum listed below up to a maximum of 4TB, in 1MB increments.

The IQ RESERVE and TEMPORARY RESERVE parameters reserve a range of blocks, so that the dbspace can be resized at a later time. Making IQ RESERVE larger than needed can use additional disk space, however.

**Table 5-3: Default and minimum sizes of IQ and Temporary Stores**

<i>IQ page size</i>	<i>Default size of IQ Store</i>	<i>Default size of Temporary Store</i>	<i>Minimum IQ Store size when specified explicitly</i>	<i>Minimum Temporary Store size when specified explicitly</i>
65536	4096000	2048000	4MB	2MB
131072	8192000	4096000	8MB	4MB
262144	16384000	8192000	16MB	8MB
524288	32768000	16384000	32MB	16MB

## Choosing a Catalog page size

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
```

## Choosing a block size for your database

In nearly all cases you should rely on the default block size, which is based on the IQ page size.

## Enabling Java in the database

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 Programmer's 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 Adaptive Server Anywhere are also supported in the IQ Catalog Store.

Sybase IQ supports access to Adaptive Server Anywhere tables from IQ, and to IQ tables from Adaptive Server 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 Adaptive Server Anywhere database, or queries to IQ tables from an Adaptive Server Anywhere database. For details on using remote data access capabilities, see Chapter 16, “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:

- “Introduction to Java in the Database” in the *Adaptive Server Anywhere Programming Guide*
- “Using Java in the Database” in the *Adaptive Server Anywhere Programming Guide*

- Appendix B, “Data Access Using JDBC”
- Appendix C, “Debugging Logic in the Database”

## Reserving space to handle out-of-space conditions

In the event that you run out of space to perform an operation, you see a message telling you to add more disk space. Before this occurs, set the options `MAIN_RESERVED_DBSPACE_MB` and `TEMP_RESERVED_DBSPACE_MB`. These options provide room for checkpoint, commit, and release savepoint operations so that you can add more space under all conditions. These options are set at run-time to accommodate half of your last dbspace up to 200MB, but may need adjusting, depending on the number of concurrent connections and your IQ page size. Do not wait until you have run out of space to set these options. See Chapter 2, “Database Options” in the *Sybase IQ Reference Manual* for option details.

## Creating databases with multiplex functionality

Before creating a multiplex database

Record the value for each dialog item in the following table.

**Table 5-4: Data required to create a multiplex database**

Dialog item	Data type/length	Notes	Value
Host name	CHAR 30	Name of the machine where the database engine will run.	
Server name	CHAR 30	Server name for the write server. (The server name must be unique across the local area network.)	
Raw device local path(s) on UNIX	CHAR 2048	Full path/ directory specification.	
Raw device name(s) on Windows	Varies	<i>PhysicalDriveN</i> (unpartitioned device) or assigned letter (partitioned device) See “Specifying an IQ PATH” on page 187 for details.	
Database path	CHAR 1024	Create the database files on a local disk, not a remote location.  The Create Database wizard asks for the path to the .DB file. Users cannot specify where the server will be started.	
Database name	CHAR 30	Database name. Included in the path.	

Dialog item	Data type/length	Notes	Value
Temporary Store name	CHAR 30	By default, IQ adds the file type <i>.IQTMP</i> .	
Overview of multiplex creation		<p>Refer to this information when you create servers and databases.</p> <p>See Chapter 8, “Physical Limitations” in the <i>Sybase IQ Reference Manual</i> for an important note about initializing raw devices on Sun Solaris.</p>	
Creating query servers		<p>For a multiplex database, create a database that has a raw device for the IQ Store. Chapter 4, “Managing Databases” in <i>Introduction to Sybase IQ</i> explains how to create IQ databases.</p> <p>To enable your database for multiplex capability, add query servers, as described in the following section.</p> <p>Once you create a database, you can create tables and indexes, which are stored in the IQ Store. For details, see “Creating tables” on page 239 and “Creating indexes” on page 255. You can also create procedures and views, which are stored in the system tables. For syntax, see the <i>Sybase IQ Reference Manual</i>.</p> <p>To set up data sources for your servers, see <i>Sybase IQ Installation and Configuration Guide</i>. You may also add dbspaces to servers as needed. For details, see “Adding dbspaces” on page 216.</p>	
		<p><b>Note</b> Sybase IQ does not support heterogeneous multiplexes (UNIX and Windows servers in a mixed multiplex). Write and query servers must be on the same hardware platform.</p>	

To activate multiplex capability, simply add one or more query servers to an IQ server.

#### ❖ Adding a query server

- 1 Start Sybase Central using the method appropriate for your platform.
- 2 Connect to a Sybase IQ server.
- 3 Open the database and double-click Multiplex.
- 4 Choose File > New > Query Server (Alt+F, N, Q).
- 5 The first time you add a query server, Sybase Central verifies the server name and the port number for the write server. Usually, you will want to keep the same name and port number that the server is already using. Sybase Central will verify that the Agent is running on the write server’s host.

- 6 The server to which you are currently connected becomes the write server for the multiplex by default when you create a query server.

If you wish to use another server for the write server, change the Write Server Name and Port Number now by highlighting and typing over them on the Specify Server Options screen.

The administrative shell scripts are provided as a convenient way to start and stop the IQ server and start the SQL Remote process from the shell. For more information, see “Using administrative shell scripts” on page 210.

To create or update the default administrative shell scripts, click the checkbox.

- 7 Click Next.
- 8 On the Set Connection Parameters screen, the Query Server Name defaults to *writeservernameNN*, where *NN* is the number of servers already configured plus one.

To change the Server Name, select and type over it.
- 9 Select the host on which the query server will run.
- 10 The Port Number defaults to the port number of the current connection plus 1. If desired, type over this number to change it.
- 11 Click Next.

- 12 To change the default, type the path to the database file, or use the Browse button below the text box to select a new path. Note that Browse is only enabled if the host specified in the previous screen is the local host. The Browse button can locate either a directory or a file.



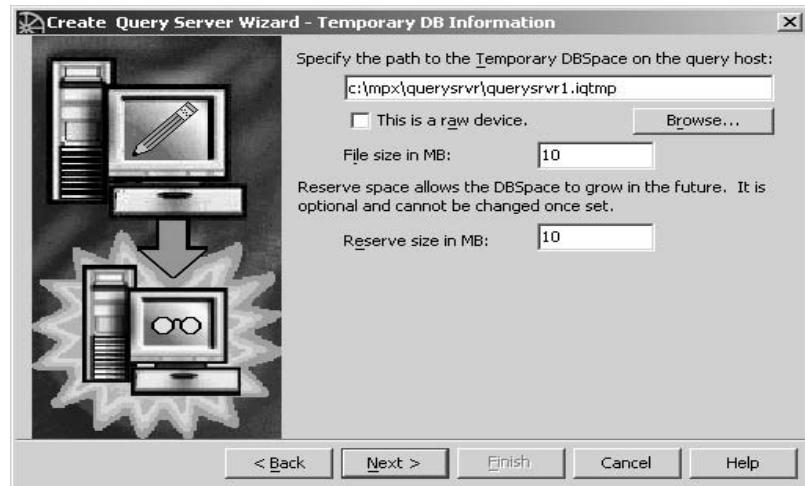
---

**Note** Open Client applications connect to IQ servers via unique labels that map to the database name. In typical multiplex installations, this can be an issue, because the default database name is the same for all servers in the multiplex. This window allows you to give the data base a unique name for each query server. These unique names may then be used to define explicit connection entries in the interface file.

---

- 13 Click Next.
- 14 Accept the default for the Temporary Store Path, or select the name and type over it if incorrect.
  - On Windows, do not supply the "\.\\" in the device name for a raw disk or partition.

- On UNIX, each platform has its own format for raw device names. Check the operating system documentation for the correct format. Sybase recommends using symbolic links to the raw partitions.



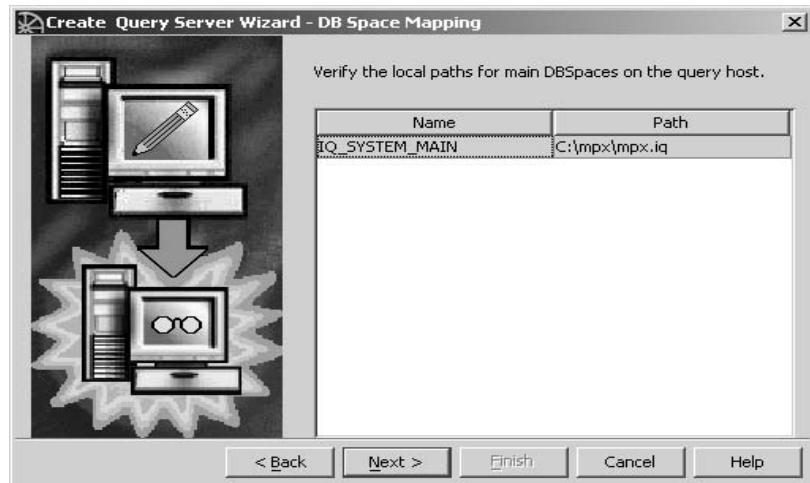
Click the checkbox if the Temporary Store is on a raw device. If it is not, an additional text box displays where you must specify the size of the Temporary Store in MB or accept the default of 10MB.

If you specify a file name without the full path, Sybase Central creates the Temporary Store under the same directory as the catalog store (.DB file).

Specify the optional Reserve Size in MB.

Click Next. Note that the Browse button is only enabled for the local host.

- 15 On the DB Space Mapping screen, check that the paths and names listed for the main dbspaces are correct as seen by the new query server's host. The main dbspaces must be available to each server in the multiplex. If one is incorrect, select the path and type over it. The paths and names default to the values at the write server.



---

**Note** You may find it convenient to arrange drive letter assignments (on Windows systems) or symbolic links (on UNIX systems) to match paths and names across the hosts in the multiplex.

---

- 16 The creation parameters for your query server display. Check the Create Connection Profile option to save the parameters in a file to simplify future connections.



- 17 Click Finish to create the query server, or Back to return to a previous screen and change information. A status box at the bottom of the screen shows the progress of create operations.

**Note** When the server is on the same host as the Sybase Central system, and this host is a Windows machine, starting, stopping, and synchronizing servers displays one or more DOS command windows on the affected machines. The DOS command window always displays on the Windows machine where the IQ Agent is performing certain actions. For example, you may see a window titled DBSPAWN.EXE when a new server is spawned in the background. The appearance and disappearance of these windows is normal and requires no action.

---

Directories and scripts installed

Multiplex creation installs several directories and files.

**Table 5-5: Contents of the write server's database directory**

Folder, Directory, or Filename	Purpose
repDirs	Directory for dbremote message passing. <i>Do not put any files in this directory or remove subdirectories underneath.</i> It contains a <i>logfiles</i> subdirectory for output messages from dbremote, and a subdirectory for each server in the multiplex.
<i>dbname.db</i>	File that contains the IQ Catalog Store
<i>dbname.iqmsg</i>	File that contains messages from Sybase IQ
<i>dbname.iqtmp</i>	File that contains the IQ Temporary Store. (The creator of the database may have placed this file in another location.)
<i>dbname.log</i>	File that contains the database transaction log.
<i>params.cfg</i>	Contains startup parameters for this particular server. May be customized for performance or specific requirements.

If you generated administrative scripts, they are created in the database directory. For more information, see “Using administrative shell scripts” on page 210.

The query server’s database directory is similar, but does not contain a *repDirs* directory.

#### Checking server status

Adding the first query server enables the database for multiplex. The multiplex folder now contains a list of servers in the multiplex. Opening the folder displays the details shown in Table 5-6.

**Table 5-6: Multiplex folder contents**

Column	Description
Name	Server name
Role	Write or Query server
Sync State	Writer, Query, Not Running (Query servers only), Synchronized or not synchronized (Query servers only)
SQL Remote	Active or Inactive
As Of	When the displayed information was last updated. For example: "8/5/04 7:01:18 PM"

You can use this display to identify which servers need to be synchronized. Icons beside the server name echo each server's Sync State, as shown in Table 5-7.

**Table 5-7: Server status icons**

Icon	Run status	Synchronization status
Green light (Top light)	Running	Synchronized
Yellow light (Middle light — Query server only)	Running	Unsynchronized
Red light (Bottom light)	Not running	Not Running

Status displays are not instantaneous and depend on network latency. To refresh the status display, select View > Refresh Folder from the main menu bar.

## Multiplex management without Sybase Central

It is possible to manipulate the Sybase IQ multiplex configuration and perform all multiplex administration tasks without Sybase Central.

All multiplex dbspaces are created, altered and dropped using the Interactive SQL statements CREATE DBSPACE, ALTER DBSPACE and DROP DBSPACE commands, first on the owning server and then on the write server. These second operations require extended dbspace syntax. Syntax and procedures are planned for a future documentation release.

To access the Sybase Product Manuals Web site for the latest documentation revisions, go to Product Manuals at <http://www.sybase.com/support/manuals/>.

## System tables and multiplex dbspace management

Multiplex configuration information is found in the system table SYSIQFILE and in the following special system tables:

- IQ\_MPX\_INFO
- IQ\_MPX\_STATUS
- IQ\_MPX\_VERSIONLIST

For details, see Chapter 9, “System Tables,” *Sybase IQ Reference Manual*.

The following rules govern dbspace management in the multiplex:

- Each multiplex server has a set of rows in SYSIQFILE. This set includes:
  - One row for each shared main dbspace
  - One row for each temporary dbspace of the server
  - One row for the .message file (*.iqmsg*)
  - If it is a query server, one row for each local dbspace of the server
- The `server_name` column of SYSIQFILE identifies the server that owns the row. Each row that belongs to the write server has the empty string, '' , as the `server_name`. Each row that belongs to a query server names that query server in the `server_name` column.

Do not confuse owning a row in SYSIQFILE with owning the dbspace. For example, a shared main dbspace named *main2* is owned by the write server, but each query server owns a row in SYSIQFILE for dbspace *main2* with the query server’s `server_name`.

- Filenames may use either absolute or relative paths. On UNIX, they may also be UNIX links rather than actual files. The most flexible setup is to use relative paths for NFS files and, on UNIX, relative paths to links for raw devices.
- Filenames used for a shared main dbspace may differ across servers, either because the device names differ on different servers or because the DBA uses links with different names for each server.

- SYSFILE contains one row for each physical IQ dbspace file. Since there is one row in SYSIQFILE for each main dbspace for each server, there is a many-to-one relationship between the rows in SYSIQFILE and the rows in SYSFILE for shared main dbspace. All temporary and local dbspaces have exactly one row in SYSFILE and one row in SYSIQFILE.
- The query server rows in SYSIQFILE for a shared main dbspace are referred to as **aliases**. The term alias may also refer to any row in SYSIQFILE that belongs to another server.

## System procedures for managing multiplex dbspaces

The following system procedures simplify multiplex dbspace management:

**Table 5-8: Multiplex stored procedures**

Procedure	Purpose
sp_iqmpxcountdbremote	Returns a count of dbremote connections for a multiplex database.
sp_iqmpxgetconnversion	Displays the version number for a specified connection.
sp_iqmpxreplacewriteserver	Converts the query server on which it runs into the new write server for the multiplex.
sp_iqmpxvalidate	Detects inconsistencies in the multiplex configuration information.
sp_iqmpxversioninfo	Displays server type (W for write server, Q for query server, S for single-node mode) and synchronization status.
sp_iqmpxcfg_<servername>	Sets up specified query server for SQL Remote replication.

For details, see Chapter 10, “System Procedures,” *Sybase IQ Reference Manual*.

## Mixed-version multiplexes

Any multiplex where all servers are not at the same version is a mixed-version multiplex. Upgrading Sybase IQ from version 12.6 to 12.7, for example, typically results in a mixed-version multiplex for a short period.

Please see “Before You Install” in the *Sybase IQ Installation and Configuration Guide* Chapter 1 for details about running multiple IQ Agents on a mixed-version multiplex.

The following notes apply to operations on mixed-version multiplexes:

- Creating a local store

Creating a local store succeeds on a 12.6 or higher query server but not on the write server and is, therefore, lost at the next synchronize.

- Creating a main store

The create operation succeeds but displays a warning. The space cannot be deleted explicitly from any 12.6 or higher query server. The query servers that still define the deleted main store will continue running, but once stopped will not restart without override flags, and a file\_id mismatch between write and query servers causes subsequent main store creates to fail.

## Synchronizing query servers

The process that updates Catalog Stores on query servers is called **synchronization**.

Synchronizing copies the write server's version of the database catalog to a query server. In general, Sybase IQ propagates DDL and DML changes to query servers, so you only need to synchronize a query server for these server management actions:

- While creating a new query server
- Restoring a query server from backup
- Restarting a query server that has been excluded
- After running the write server in single-node mode

The Sybase IQ multiplex feature automatically makes committed changes in the contents of IQ tables visible on all servers in the multiplex. Schema changes include adding, deleting, or changing tables, user IDs, indexes, and views.

**Before you synchronize**

To specify user logins, permissions or set options unique to a particular query server, instead of duplicating the settings on the write server, define a per-server stored procedure on the write server to run automatically after the query server is synchronized. The stored procedure must be named *sp\_mpxcfg\_<servername>* where *<servername>* is the name of the query server. The procedure must be owned by user DBA.

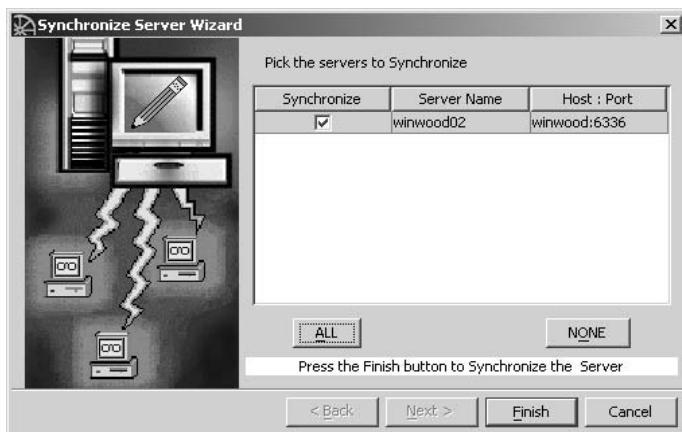
The following statements are an example of those you could add to *sp\_mpxcfg\_<servername>*. For a complete list of database options, see the *Sybase IQ Reference Manual*.

```
SET OPTION PUBLIC.DATE_FORMAT = 'Mmm dd yyyy';
SET OPTION ANSINULL = 'OFF';
SET OPTION CONVERSION_ERROR = 'OFF';
GRANT CONNECT TO fiona IDENTIFIED BY password;
```

**❖ Synchronizing query servers**

- 1 Make sure that the write server is running. This is required for synchronization to succeed.
- 2 If not already connected, connect to any server in the multiplex. (For details, see “Opening the Connect dialog” on page 77.)
- 3 Double-click the database to open it.
- 4 Click the Multiplex folder.
- 5 To synchronize an individual server, right-click that server and choose Synchronize. You can keep running queries on the other query servers.

To synchronize all the servers in the multiplex, click All (Alt+A). To synchronize some servers, check the box beside each server name.



Click Finish to synchronize or cancel to quit. Sybase Central briefly stops each query server, replaces its Catalog Store, then restarts that query server. The write server continues running throughout the operation.

## Persistent objects

Users may create an IQ Local Store on one or more query servers in order to create and store persistent objects. These objects will be visible to all users of that query server, but not to users on other servers of the multiplex, so users of that server could continue development operations without affecting the shared main storage.

Users can create, modify and query the following persistent objects and associated metadata, using SQL statements:

- Base tables
- Indexes
- Domains (user-defined data types)
- Referential integrity constraints
- User names and permissions
- Views
- Stored procedures

For details about creating and populating persistent objects, see Table 5-9 on page 241.

---

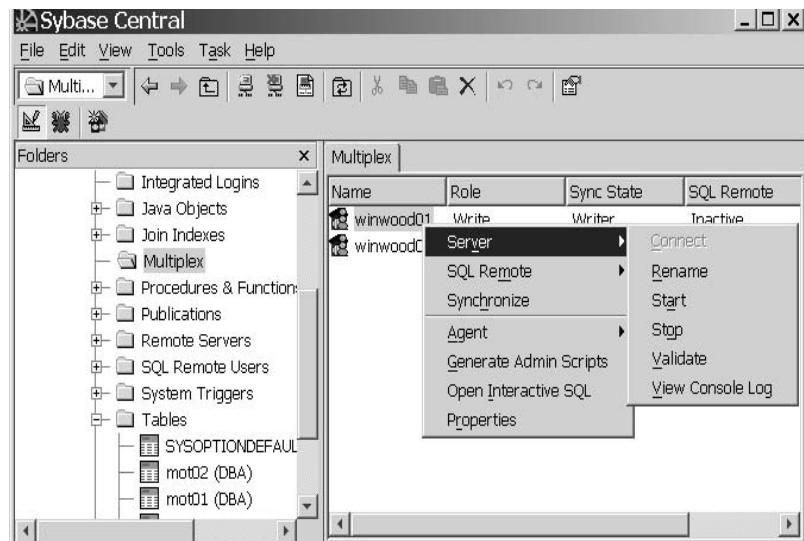
**Note** DROP DOMAIN is prohibited on a multiplex query server without a local IQ Store. Synchronizing the multiplex removes domains from query servers without IQ Local Stores. If the Query Server has an IQ Local Store, then both CREATE DOMAIN and DROP DOMAIN are permitted.

---

## Managing multiplex servers

To manage your servers, open the database Multiplex folder and right-click on the desired server in the right pane. The icon beside each server's name indicates its included/excluded status (see Table 5-7).

The following popups display when you select a write server:



You can perform the following operations from the popup menus:

- Connect, rename, start, stop, or validate the selected server.
- Start or stop SQL Remote for that server.
- Synchronize the selected server or all servers.

- View IQ Agent properties (version, port number, timeout setting)
- Delete or exclude servers from the multiplex (query servers only)
- Generate Administrative Scripts
  - Use this option to create or update the optional scripts that start or stop the server, synchronize query servers and start the SQL Remote process. You should generate the scripts after you install a new release or update of Sybase IQ.
- Open Interactive SQL
  - This option connects without requiring the user to enter login information.
- View the selected server's properties

---

**Note** Wizards on the write server (Start Server, Stop Server, and Synchronize) allow you to affect all servers in the multiplex.

---

## Including and excluding servers

For excluded servers, the menu options Start, Stop, and Synchronize are dimmed. When you include a query server again, the Synchronize Query Server wizard opens. After you synchronize the server, all menu options are restored.

If the server is included, the menu option is exclude, and vice versa. Include/exclude does not apply to the write server.

Excluded servers are not updated during synchronize or replicate operations, and the write server will not track their table versions. If you exclude a server instead of deleting it, you can preserve its configuration.

## Validating the multiplex

The validate option checks the multiplex and notifies you of errors that would prevent some or all servers from starting in multiplex mode. You can run it on any server in the multiplex environment.

This menu option only reports the *level of severity* of errors found. (You need to check the log file in order to read the errors. See “start\_asiq log file” on page 60.) The levels of severity are:

- 0 — no configuration errors

- 1 — dynamic state is not as expected (e.g., dbremote process not running)
- 2 — non-fatal configuration error (multiplex operation impaired)
- 3 — fatal configuration problem (one or more servers may not start)

When a server starts, it writes the following message to its server log if an ERROR or FATAL level problem is detected:

```
Multiplex environment incorrect for this server  
Please connect and run procedure sp_iqmpxvalidate for  
help.
```

The validate option runs the system procedure `sp_iqmpxvalidate`, which performs multiple checks on the SQL Remote configuration and on the tables `SYS.SYSIQFILE` and `DBA.IQ_MPX_INFO`. For details about these tables, see “`SYSIQFILE` system table” and “`IQ_MPX_INFO` system table” in *Sybase IQ Reference Manual Chapter 9, “System Tables.”*

If called interactively with the calling argument 'Y,' the `sp_iqmpxvalidate` procedure also returns a full list of the errors it can find.

The possible error messages the `sp_iqmpxvalidate` procedure can return are:

- **FATAL**—

```
FATAL: IQ_MPX_INFO and/or IQ_MPX_STATUS table missing
```

```
FATAL: Missing procedure sp_iqevbegintxn, sp_iqmpxversionfetch,  
sp_iqmpxpostsyncqueryserver, sp_iqmpxcountdbremote,  
sp_iqmpxsubscribeuser, sp_iqmpxunsubscribeuser, sp_iqmpxsetpublisher,  
sp_iqmpxprotectexec, or sp_iqmpxaddremoteusers
```

```
FATAL: Missing trigger tr_iqmpx_tlv_stat or tr_iqmpx_node_stat
```

```
FATAL: Missing event ev_iqbegintxn or ev_iqopendb
```

- **ERROR**—

```
ERROR: rowcounts of DBA.IQ_MPX_INFO and DBA.IQ_MPX_STATUS differ  
ERROR: SYS.SYSIQFILE configured for multiplex but DBA.IQ_MPX_INFO is  
empty
```

```
ERROR: Server counts in DBA.IQ_MPX_INFO and SYS.SYSIQFILE differ  
ERROR: server_name columns of DBA.IQ_MPX_INFO and DBA.IQ_MPX_STATUS  
mismatch
```

```
ERROR: DBA.IQ_MPX_INFO must have exactly one write server
```

```
ERROR: Invalid value in column DBA.IQ_MPX_INFO.role
```

```
ERROR: Invalid value in column DBA.IQ_MPX_INFO.role
```

```
ERROR: Query server names in DBA.IQ_MPX_INFO and SYS.SYSIQFILE differ
```

```
ERROR: Remote user <user> for server <server> is not a valid user
```

```
ERROR: DBA.IQ_MPX_INFO.remote_user is not the publisher for this server
```

```
ERROR: publisher configuration is incorrect
ERROR: IQMP_PUB publication is missing
ERROR: DBA.IQ_MPX_INFO.remote_user for the write server has remote
privileges
ERROR: REMOTE privileges incorrect for DBA.IQ_MPX_INFO.remote user for
some query server
ERROR: DBA.IQ_MPX_INFO.remote_user for some active query server is not
subscribed to IQMP_PUB
ERROR: REMOTE privileges incorrect for DBA.IQ_MPX_INFO.remote_user for
the write server
ERROR: DBA.IQ_MPX_INFO.remote_user for the write server is not subscribed
to IQMP_PUB
ERROR: Main dbspace counts for server <server>, write server mismatch in
SYS.SYSIQFILE
ERROR: Server <server> has no Temp dbspaces
ERROR: Server <server> has no IQ message file
ERROR: parameter mismatch between servers for shared dbspaces in
SYS.SYSIQFILE
ERROR: Write server inn DBA.IQ_MPX_INFO matches query server dbspaces in
SYS.SYSIQFILE
```

- **WARNING—**

```
WARNING: DBA.IQ_MPX_INFO has write server but no query servers
WARNING: Server counts in DBA.IQ_MPX_NFO and SYS.SYSIQFILE differ.
WARNING: Main dbspace counts for server <server>, write server mismatch
in SYS.SYSIQFILE [note: warning if indicated server has more dbspaces]
WARNING: current_version < current version in DBA.IQ_MPX_STATUS
WARNING: query server oldest_version > current_version in
DBA.IQ_MPX_STATUS
WARNING: SQL Remote (dbremote) is not running on this server
WARNING: catalog_version for query server newer than write server in
DBA.IQ_MPX_STATUS
```

## Using administrative shell scripts

If you generated administrative shell scripts for your servers, these scripts are located in the database directory. (On a Windows host, these scripts are *.bat* files. On UNIX, they are shell scripts.) Open each script with a text editor to see comments about how to use it.

Scripts that require a password define it as an input parameter.

These scripts may be used to automate operations that you would normally perform periodically with Sybase Central. The scripts can:

- Start and stop servers

- Start dbremote
- Start dbisql
- Synchronize query servers

For example, you could use the scripts to synchronize each query server during the night.

UNIX systems	You can use the crontab system utility to schedule the administrative tasks.
Windows systems	You will need a task scheduler application that can start up specified processes with specified permissions at certain times of the day. Install this application on each machine in the multiplex.
Using scripts to synchronize servers	Synchronizing query servers requires actions on two machines of the multiplex. Read the documentation within each script to make certain that you run them in the proper order.

## **Viewing server properties**

Choose Properties. The property sheet lists properties on three tabs: General, Extended Information, and Options. Click Help on each tab for details about the properties listed.

## **Deleting query servers**

You can only delete query servers, not the write server. To delete a query server, right-click on that server and choose Delete from the popup menu.

Currently, the delete will fail if another user is logged in as DBA for a database on the specified write server.

The wizard prompts you to verify the delete operation before continuing.

Click the option button to delete associated files, if desired, and choose Finish.

If the wizard fails to remove any of the directory paths associated with the server, it displays a notification message. This is typically due to a sharing violation. The delete operation succeeds and containers redisplay as appropriate. You should then drop any files/paths that could not be dropped automatically.

## Updating multiplex databases

The following rules govern DDL operations on multiplex databases:

- You can create and populate Adaptive Server Anywhere tables and IQ Temporary tables, but on query servers, the effect of these operations is lost the next time you do a synchronize operation if the query server does not have an IQ Local Store.
- If you create objects in the SYSTEM dbspace, they are created in the Catalog Store, an Adaptive Server Anywhere (ASA) store. Anything created in SYSTEM on a query server is lost the next time you synchronize.
- Create or drop triggers or events *on the write server only*.

## Resolving static collisions

A **static collision** is a database state that prevents a DDL statement from executing, even if the statement is retried. For example, attempting to drop a user data type fails if that data type is in use in a table definition. This situation is static, because retrying the statement never succeeds, even after all users are disconnected or after the server is restarted. In an interactive setting, a static collision causes a SQL error.

A **dynamic collision** is a database server state that prevents a DDL statement from executing at the moment, but eventually clears on its own and allows the statement to succeed, if the statement is retried. For example, altering a table fails if the table is in use in a query by another user. This situation is dynamic, because retrying the statement succeeds once other operations complete, after all users are disconnected, or after the server is restarted. In an interactive setting, a dynamic collision results in a SQL error.

In a multiplex environment, the query server resolves a static collision that occurs while executing a DDL statement propagated from the write server by renaming objects that conflict. The object (a table or domain) is given a new name with the character “~” (tilde) appended, followed optionally by a sequence number, to create a unique name that is unlikely to look like a name users would choose.

For example, suppose that you create a table *employees* on the write server, and a query server persistent table with the same name already exists (created on a local store in the query server), then Sybase IQ renames the query server table *employees~1*.

For information on objects that have been renamed due to static collisions, run the stored procedure `sp_iqmpxrenameinfo`. This procedure returns both the old name and the new name of the object.

To resolve a dynamic collision during DDL execution, the query server finds the connections responsible and disconnects them.

The disconnect is logged in the `.iqmsg` file. Here is an example of a message that appears in the `.iqmsg` when a dynamic collision occurs:

```
Shared IQ Store update DDL statement:  
drop table DBA.gtt44  
Disposition: SQLSTATE:42W21 --  
dropped 1 connection(s) for table:  
DBA.gtt44 Retry successful
```

Users can avoid collisions by managing domain and table names carefully across the members of the multiplex. To avoid the inconvenience of disconnects, schedule DDL at the write server that affects normal query operation on the multiplex servers to avoid active users.

## Setting database options

Database options are configurable settings that change the way the database behaves or performs. In Sybase Central, all of these options are grouped together in the Set Options dialog. In Interactive SQL, you can specify an option in a SET OPTION statement.

- ❖ **Setting options for a database (Sybase Central)**
  - 1 Open the desired server.
  - 2 Right-click the desired database and choose Set Options from the popup menu.
  - 3 Edit the desired values.
- ❖ **Setting options for a database (SQL)**
  - Specify the desired properties within a SET OPTION statement.

---

### Tips

With the Set 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.

---

For more information, see Chapter 2, “Database Options.” in the *Sybase IQ Reference Manual*.

## Showing system objects in a database

In a database, a table, view, stored procedure, or domain is a system object. System tables store information about the database itself, while system views, procedures, and domains largely support Sybase Transact-SQL compatibility.

In Interactive SQL, you cannot show a list of all system objects, but you can browse the contents of a system table.

❖ **Showing system objects in a database (Sybase Central)**

- 1 Open the desired server.
- 2 Right-click the desired connected database and choose Filter Objects by Owner.
- 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).

❖ **Browsing system tables (SQL)**

- 1 Connect to a database.
- 2 Execute a SELECT statement, specifying the system table you want to browse. The system tables are owned by the SYS user ID.

**Example**

Show the contents of the table sys.systable in the Results pane.

```
SELECT *
FROM SYS.SYSTABLE
```

## Disconnecting from a database

When you are finished working with a database, you can disconnect from it. Sybase IQ also gives you the ability to disconnect other users from a given database; for more information about doing this in Sybase Central, see *Introduction to Sybase IQ*.

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' )
```

### ❖ Disconnecting from a database (Sybase Central)

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

### ❖ Disconnecting from a database (SQL)

- Execute a `DISCONNECT` statement.

#### Example 1

The following statement shows how to use `DISCONNECT` from Interactive SQL to disconnect all connections:

```
DISCONNECT ALL
```

#### Example 2

The following statement shows how to use `DISCONNECT` in Embedded SQL:

```
EXEC SQL DISCONNECT :conn_name
```

### ❖ Disconnecting other users from a database (SQL)

- 1 Connect to an existing database with DBA authority.
- 2 Execute a `DROP CONNECTION` statement.

#### Example

The following statement drops the connection with ID number 4.

```
DROP CONNECTION 4
```

For more information, see “`DISCONNECT` statement [DBISQL]” and “`DROP CONNECTION` statement” in *Sybase IQ Reference Manual*.

## 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 pathname *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.

## Working with dbspaces

A DBA can determine which tables and indexes reside on a given dbspace, relocate objects to other dbspaces, and drop any dbspace after emptying it of data. A DBA can also define the number of writes to each dbspace before the disk striping algorithm moves to the next stripe.

See the section “Dbspace management example” on page 228 for examples of using the dbspace SQL statements and stored procedures to create dbspaces with reserve space, modify the size of dbspaces, display information about dbspaces, relocate objects, and drop empty dbspaces.

## Adding dbspaces

When you create a database, it has only one file for storing permanent IQ data, one file for storing Catalog data, and one file each for the IQ message log and the Temporary Store. Each of these files is a dbspace, as described in “Creating a database with SQL” on page 183. Initially, the definitions of all IQ database objects go into the SYSTEM dbspace (the Catalog Store), and all IQ data is placed in the IQ\_SYSTEM\_MAIN dbspace (the IQ Store).

Each dbspace has a maximum size of 4096GB (4TB) on raw devices. The maximum size for operating system files varies by platform; see Chapter 8, “Physical Limitations” in the *Sybase IQ Reference Manual* for details. On some platforms you must enable large file system files to reach the maximum size.

The catalog file size maximum for all platforms is 1TB, except for Windows systems with FAT 32 file systems, which have a 4GB limit. Windows systems with NTFS support the 1TB maximum.

You can only specify SIZE and RESERVE for the IQ Store and IQ Temporary Store, not for the Catalog Store.

In the large databases typical of a data warehouse, you will need to add dbspaces to any database. You create a new database file—a dbspace—using the CREATE DBSPACE statement, or the Sybase Central Add Dbspace Wizard. A new dbspace can be on the same or a different disk drive as the existing dbspaces. You must have DBA authority to create new dbspaces.

When you create a new dbspace, it contains no user data. When you create tables and indexes and load them, if DISK\_STRIPING is on, Sybase IQ distributes the data as equally as possible among any existing dbspaces that are not already full. This technique optimizes performance.

Because Sybase IQ fills dbspaces in this way, you cannot specify that a particular IQ table be loaded into a particular dbspace. You can only indicate the IQ Store as the dbspace IQ\_SYSTEM\_MAIN, and the Temporary Store as the dbspace IQ\_SYSTEM\_TEMP. The only way to control the location of a table within the IQ Store is to make all other dbspaces read-only (using the ALTER DBSPACE command) while loading.

---

**Note** This behavior differs from that of Adaptive Server Anywhere, which allows you to place tables in a particular dbspace.

---

#### How the number of dbspaces affects resource use and performance

The maximum number of dbspaces, or files, 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. However, you should never allow a situation where you come close to the maximum.

Increasing the number of dbspaces has no real impact on memory use or performance.

---

**Note** On HP and AIX platforms, your use of overlapped I/O improves when you divide data among more dbspaces.

---

When data is stored on raw partitions, you can have one dbspace per drive. See Chapter 8, “Physical Limitations” in the *Sybase IQ Reference Manual* for an important note about initializing raw devices on Sun Solaris.

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. In any case, your stripe size should be several times your IQ page size. IQ can also be configured to perform software striping.

For more information on disk striping and use of multiple dbspaces, see “Balancing I/O” in *Sybase IQ Performance and Tuning Guide*.

Before adding any more dbspaces you may want to estimate your space requirements. See “Estimating space and dbspaces required” for details of how to estimate space. 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* in the file *library.iq* in the same directory as the IQ\_SYSTEM\_MAIN dbspace:

```
CREATE DBSPACE library  
AS 'library.iq'
```

**Creating a dbspace in Sybase Central**

❖ **Creating a dbspace in Sybase Central**

The following steps describe how to add a dbspace to a non-multiplex database. To add a dbspace to a multiplex, skip to the next procedure.

- 1 Connect to the database.
- 2 Click the Dbspaces folder for that database.
- 3 From the menu bar, choose File > New > DBSpace.
- 4 Enter the dbspace name.
- 5 Click the type of data to be stored in this Dbspace: Main IQ Store, Temporary IQ Store or Catalog Store.
- 6 Click Next.
- 7 Enter the file path and, optionally, the size of the dbspace.
- 8 Enter the amount of space to reserve, if you expect to resize the partition at a later time.
- 9 Click Finish to create the dbspace.

Adding a main  
dbspace to a multiplex

❖ **Adding a main dbspace to a multiplex**

Before you add a main dbspace, make sure that the write server is running. It is recommended that all servers be running whenever you add dbspaces so that all paths to the new dbspace can be checked.

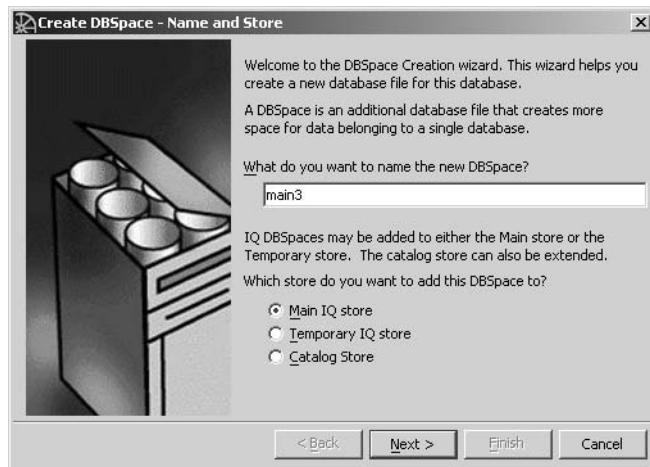
---

**Note** When creating a main dbspace on a write server, you must create aliases for the query servers before you synchronize them or they cannot open the new file.

---

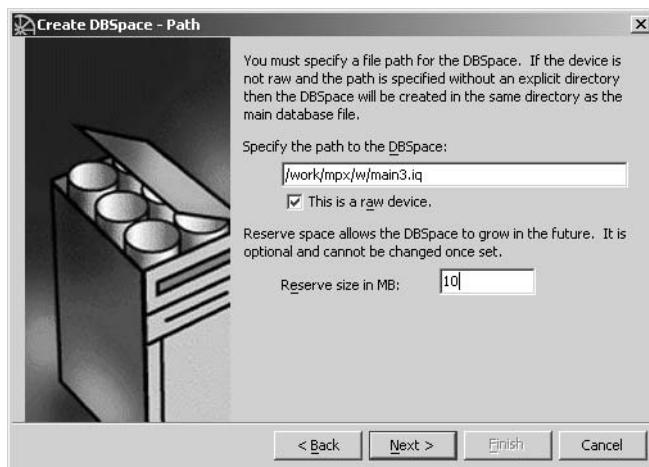
- 1 Connect Sybase Central to the multiplex write server.
- 2 Open the DBSpaces folder.
- 3 Choose File > New > DBSpace (or Alt+F, N, D), and run the Create DBSpace wizard.
- 4 On the Name and Store screen, type the name of the dbspace in the text box.

The dbspace name, an internal name for the dbspace, must be unique in a database. Dbspace names are case sensitive for databases created with CASE RESPECT. (To view the CaseSensitive setting, right-click the database, choose Properties, and view the Extended Information tab.)



On a write server, the DBSpace wizard creates a dbspace in the IQ Store (Alt+M) by default. If you prefer the IQ Temporary Store, click the Temporary IQ Store option (Alt+T).

- 5 Click Alt+N to continue or Cancel to quit.
- 6 On the Path screen, type the dbspace path in the text box (Alt+D). If the device is not raw and the path is specified without an explicit directory, then IQ creates the dbspace in the same directory as the main database file. You may specify the filename suffix .iq for an IQ Store.



- 7 The raw device option is selected by default (Alt+a). Tab or click Alt+a and type the size in MB of reserve space to allow for future growth. Reserve space is optional and cannot be changed once set.

If your dbspace is a file, deselect the raw device option (Alt+a). Type the file size value (number of MB) in the text box (Alt+i), or accept the 10MB default and tab (or Alt+e) to the Reserve size box.

For an IQ Store, you now need to define the path for the new dbspace on each query server. Sybase Central adds the necessary system table entries to make the new main dbspaces visible to all query servers. Sybase Central displays the DB Space Mapping screen with a path for each server. Edit them as needed so that each server uses the correct filename to access the new IQ Store.

- On Windows, do not supply the “\\.\.” in the device name for a raw disk or partition. Ensure that the drive letters or Physical Drive numbers you provide refer to the same raw partition or disk in the shared disk array.
- On UNIX, each platform has its own format for raw device names. Check the operating system documentation for the correct format or use symbolic links to the raw partitions.

If device names are different on different nodes and the nodes are UNIX systems, you may use symbolic links to the differing raw devices. These symbolic link names can then be used as pathnames for the dbspaces on the various servers.

For example, if a UNIX write server has a shared raw device `/dev/rdsk/c0t3d0s1` which the query server sees as device `/dev/rdsk/c1t3d0s1`, you may optionally create a symbolic link when adding this to the main IQ Store on the write server:

```
ln -s /dev/rdsk/c0t3d0s1  
/wserver_home/main02
```

Add a corresponding symbolic link for the query server:

```
ln -s /dev/rdsk/c1t3d0s1 /qserver_home/main02
```

This will allow both servers to see the new dbspace using the relative filename `main02`. Use this as the filename for adding the dbspace to the write server, and leave the optional size field blank since IQ will deduce the proper size for a raw device. The query server can use the same relative filename, `main02`.

See *Sybase IQ Troubleshooting and Recovery Guide* for information on correcting dbspace count mismatches between write and query servers.

- 8 Click Finish (Alt+F) to create the dbspace.

#### Adding a temporary dbspace to a multiplex

##### ❖ Adding a temporary or local dbspace to a multiplex

You may add one or more temporary and/or dbspaces to multiplex servers as needed.

- 1 Connect Sybase Central to a write or query server. (To add a local dbspace, you must connect to a query server.) The write server must be running and available.
- 2 Choose File > New > DBSpace (or Alt+F, N, D) or click the Add New DB Space button on the Dbspaces tab. Type the dbspace name and click Next.

You should add temporary dbspaces while the database is running in multiplex mode and the write server is running. Dbspace names and file names must be unique across the multiplex, so you may wish to encode the server name into the dbspace and file names you choose for the temporary dbspaces.

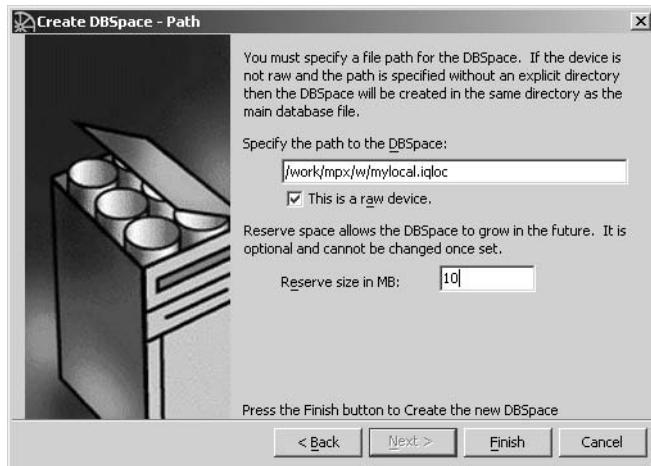
- 3 On a query server, the wizard creates a dbspace in the IQ Temporary Store by default. To create it in a IQ Local Store, click that option (Alt+L).



Click Next to accept the default, “Store IQ temporary data.”

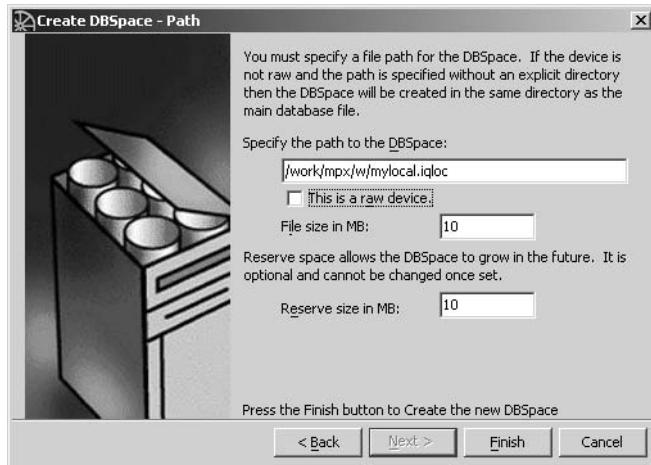
- 4 Enter the amount of space to reserve, if you expect to resize the partition at a later time.
- 5 Type the path of the dbspace. You may specify the file extension *.iqtmp* for IQ Temporary Store, or *.iqloc* for an IQ Local Store.

- 6 On the Path screen, type the dbspace path in the text box (Alt+D). If the device is not raw and the path is specified without an explicit directory, then IQ creates the dbspace in the same directory as the main database file. An IQ Store may have any file extension.



- 7 The raw device option is selected by default (Alt+a). Tab or click Alt+a and type the size in MB of reserve space to allow for future growth. Reserve space is optional and cannot be changed once set.

If your dbspace is a file, deselect the raw device option (Alt+a). Type the file size value (number of MB) in the text box (Alt+i), or accept the 10MB default and tab (or Alt+e) to the Reserve size box.



Issuing checkpoints  
for cleaner recovery

After you add or drop a dbspace, it's a good idea to issue a CHECKPOINT. In the event system recovery is needed, it begins after the most recent checkpoint.

## Dropping a dbspace

You can use Sybase Central or issue a DROP DBSPACE command to remove a database file. (For multiplex databases, always use Sybase Central to remove a dbspace.) In order to drop a dbspace, the following must be true:

- It must not contain any data from user tables. Sybase IQ does not allow you to drop a dbspace unless it is empty.
- It must not be SYSTEM or IQ\_SYSTEM\_MSG. These dbspaces can never be dropped.

In order to empty a dbspace, you must:

- Relocate or drop all tables on the dbspace.
- Commit or roll back only transactions that are using older versions of tables (on a multiplex server, this includes query server transactions).

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. You need to relocate the tables by using the ALTER DBSPACE command or Sybase Central to set the dbspace in relocate mode and then run the sp\_iqrelocate stored procedure.

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; see Chapter 10, “Transactions and Versioning” for details.

sp\_iqrelocate does not automatically commit. You must commit the changes before they are persistent. Also note that sp\_iqrelocate is used with main and local dbspaces only.

A DBA can determine in which dbspace tables and indexes are located by running the stored procedures sp\_iqspaceinfo, sp\_iqdbspaceinfo, and sp\_iqindexinfo. These procedures show the number of blocks used by each table and index in each dbspace.

To find out whether you can drop a particular dbspace, run the stored procedure `sp_iqdbspace`. Look at the Block Types column, which tells you the contents of each dbspace. A dbspace can be dropped if it contains only block types ‘H,’ ‘F,’ ‘R,’ ‘B,’ ‘X,’ and ‘C.’

Block type ‘A’ is data from active table versions. Use `sp_iqdbspaceinfo` to determine which tables need to be relocated. You can relocate by setting the dbspace mode to relocate (using the `ALTER DBSPACE` command), then using the `sp_iqrelocate` stored procedure.

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.

The following is sample output from `sp_iqdbspace`.

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_MAIN	D:\IQ\ dbspacedb.iq	MAIN	RW	24	10M	100M	8K	1H,64F, 32D,62 A,20X, 128M	1	1280
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RW	9	10M	20M	8K	1H,32F, 56A, 19X	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	12	10M	40M	8K	1H,32F, 59A, 49X	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 12A, 20X	1	1280

For more information on the values of the output fields of `sp_iqdbspace`, see “`sp_iqdbspace` procedure” in Chapter 10, “System Procedures” of the *Sybase IQ Reference Manual*.

## Dropping dbspaces from a multiplex database

The following procedures describe how to drop multiplex dbspaces using Sybase Central.

❖ **Dropping a local dbspace from a multiplex database**

If your database has multiple local dbspaces, follow this procedure to drop all but the last remaining local dbspace. When there is only one local dbspace, follow the steps in “Dropping the last local dbspace on query server.”

- 1 Connect to the multiplex query server.
- 2 Open the dbspaces container.
- 3 Right-click the local dbspace and choose Properties.
- 4 On the Usage tab, choose Relocate from the Mode dropdown list. Click Apply and OK.
- 5 Use `sp_iqrelocate` to relocate all data on the dbspace, then commit.
- 6 Use `sp_iqdbspace` to make sure no one is using the dbspace.
- 7 Right-click the local dbspace and choose Delete.  
If the delete dbspace fails, issue a checkpoint and repeat this step.

❖ **Dropping the last local dbspace on query server**

When a dbspace is created, IQ needs to save some checkpoint information for the transaction control. If the dbspace is the only one for the type of store (IQ Local Store, in this case), the checkpoint data is saved in the very same dbspace. This makes the dbspace “in use,” even though nothing has been done yet from a user’s point of view. The dbspace is therefore protected from being dropped.

To release the checkpoint information saved in the dbspace, a DBA needs to issue a `CHECKPOINT` statement to force IQ to flush out the checkpoint data. If there are multiple connections to the server, only the `CHECKPOINT` statement — which executes *after* the completion of all transactions temporarily overlapping with the transaction of creating the dbspace — will flush the checkpoint data. Therefore a DBA may have to issue `CHECKPOINT` several times to drop the dbspace.

A stored procedure, `sp_iqdbspace`, reports the current status of all IQ dbspaces, including whether or not a dbspace is ready to be dropped. See “Dropping a dbspace,” in Chapter 5, “Working with Database Objects” in *Sybase IQ System Administration Guide* for details on the `sp_iqdbspace` returned results.

If the IQ Local Store to be dropped is not empty, you must do the following before you can drop it:

- 1 Alter the database to relocate mode. For details, see “Altering dbspaces” in Chapter 6, “Managing Dbspaces” in *Introduction to Sybase IQ*.

- 2 Run `sp_iqrelocate` on the database. For details, see Chapter 9, “System Procedures” in *Sybase IQ Reference Manual*.

For more information on dropping dbspaces, see “DROP statement” in Chapter 6, “SQL Statements” in *Sybase IQ Reference Manual*.

❖ **Dropping a main dbspace from a multiplex database**

If you have created multiple main dbspaces, you can drop the `IQ_SYSTEM_MAIN` dbspace as long as you do not drop the last main dbspace remaining.

- 1 Connect to the multiplex write server.
- 2 Open the dbspaces container.
- 3 Right-click the main dbspace and choose Properties.
- 4 On the Usage tab, choose read-only from the Mode dropdown list. Click Apply and OK.
- 5 In Interactive SQL, run `sp_iqtransaction`. Check the `version_ID` column and make sure that no query servers are running with old version IDs. You must make sure that transactions on the query servers have caught up to the current state of the main dbspace. Check the Version ID column to make sure that none are running with old version IDs.
- 6 Reopen the dbspaces container on the write server.
- 7 Right-click the main dbspace and choose Properties.
- 8 On the Usage tab, choose Relocate from the Mode dropdown list. Click Apply and OK.
- 9 Use `sp_iqrelocate` to relocate all data on the dbspace, then commit.
- 10 After all objects and user data are relocated to other dbspaces, the dbspace is ready to drop using the `DROP DBSPACE` command. Before executing the `DROP DBSPACE` command, check the block type information on the dbspace Properties tab to be sure the dbspace does not have blocks of type ‘A’ or ‘O’ (blocks with data from active table versions or old versions that may still be in use). (You may instead run `sp_iqdbspace`, as shown in “Dbspace management example” on page 228.)
- 11 Right-click the main dbspace and choose Delete.  
If the delete dbspace fails, issue a checkpoint and repeat this step.

❖ **Dropping a temporary dbspace from a multiplex database**

- 1 Open the dbspaces container.

- 2 Right-click the temporary dbspace and choose Properties.
- 3 On the Usage tab, choose Relocate from the Mode dropdown list. Click Apply and OK.
- 4 Use sp\_iqdbspace to make sure no one is using the dbspace.

Note: You may have to disconnect users or possibly restart the server to free all temporary space.
- 5 Make sure the write server is running. (If it is not, the dbspace will appear to return after you synchronize query servers.)
- 6 Connect to the server that owns the dbspace.
- 7 Open the dbspaces container.
- 8 Right-click the temporary dbspace and choose Delete.

## 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 the following CREATE DATABASE statement:

```
CREATE DATABASE 'D:\IQ\dbspacedb'  
IQ PATH 'D:\IQ\dbspacedb.iq'  
IQ SIZE 10  
IQ RESERVE 100  
TEMPORARY SIZE 10  
TEMPORARY RESERVE 10  
JAVA OFF  
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 as  
'D:\IQ\dbspacedb.iq2'
```

```

        SIZE 10 RESERVE 20;

CREATE DBSPACE dbspacedb3 as
'D:\IQ\dbspacedb.iq3'
SIZE 10 RESERVE 40;

```

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

Use the sp\_iqdbspace system stored procedure to display information about all dbspaces in the dbspacedb database:

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_MAIN	D:\IQ\dbspacedb.iq	MAIN	RW	24	10M	100M	8K	1H,64F, 32D,62 A,20X, 128M	1	1280
dbspacedb2	D:\IQ\dbspacedb.iq2	MAIN	RW	9	10M	20M	8K	1H,32F, 56A, 19X	1045440	1046719
dbspacedb3	D:\IQ\dbspacedb.iq3	MAIN	RW	12	10M	40M	8K	1H,32F, 59A, 49X	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqttmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 12A, 20X	1	1280

Use the sp\_iqdbspaceinfo system stored procedure to display information about object placement and space usage for a specific dbspace:

```
sp_iqdbspaceinfo IQ_SYSTEM_MAIN;
```

dbspace_name	Object	MinBlk	MaxBlk	ObjSize	DBSpSz
IQ_SYSTEM_MAIN	t1	82	125	40K	10M
IQ_SYSTEM_MAIN	t1.DBA.ASIQ_IDX_T429_C1_FP	109	322	136K	10M

<b>dbspace_name</b>	<b>Object</b>	<b>MinBlk</b>	<b>MaxBlk</b>	<b>ObjSize</b>	<b>DBSpSz</b>
IQ_SYSTEM_MAIN	t1.DBA.t1c1hg	127	305	152K	10M
IQ_SYSTEM_MAIN	t2	84	107	32K	10M
IQ_SYSTEM_MAIN	t2.DBA.ASIQ_IDX_T430_C1_FP	126	321	136K	10M

Use the `sp_iqindexinfo` system stored procedure to display object placement and space usage for a specific table or index:

```
sp_iqindexinfo 'table t2';
```

<b>Object</b>	<b>dbspace_name</b>	<b>ObjSize</b>	<b>DBSpPct</b>	<b>MinBlk</b>	<b>MaxBlk</b>
t2	IQ_SYSTEM_MAIN	32K	1	84	107
t2	dbspacedb2	160K	2	1045495	1045556
t2	dbspacedb3	8K	1	2090930	2090930
t2.DBA.ASIQ_IDX_T430_C1_FP	IQ_SYSTEM_MAIN	136K	2	126	321
t2.DBA.ASIQ_IDX_T430_C1_FP	dbspacedb3	152K	2	2091032	2091053
t2.DBA.t2c1hng	dbspacedb2	136K	2	1045537	1045553

For the full syntax of the `sp_iqdbspace`, `sp_iqdbspaceinfo`, and `sp_iqindexinfo` system stored procedures, see Chapter 10, “System Procedures” in the *Sybase IQ Reference Manual*.

#### Changing the size of a dbspace

The `ALTER DBSPACE` commands in this section show you how to change the dbspace size, if necessary.

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;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_MAIN	D:\IQ\dbspacedb.iq	MAIN	RW	12	20M	90M	8K	1H,64F, 32D,62A ,128M	1	2560

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RW	7	10M	20M	8K	1H,32F, 56A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	8	10M	40M	8K	1H,32F, 59A	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 32A	1	1280

Note that if the dbspacedb database is not created with an IQ RESERVE value, the dbspace cannot be extended. The dbspace can be made smaller, however, and the size taken away from the dbspace is added to the reserve.

The IQ\_SYSTEM\_MAIN dbspace is now 20MB in size. This dbspace can be resized to 15MB using the ALTER DBSPACE command with the SIZE parameter:

```
ALTER DBSPACE IQ_SYSTEM_MAIN SIZE 15mb;
```

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_MAIN	D:\IQ\ dbspacedb.iq	MAIN	RW	15	15M	95M	8K	1H,64F, 32D,62A, 128M	1	1920
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RW	7	10M	20M	8K	1H,32F, 56A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	8	10M	40M	8K	1H,32F, 59A	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 32A	1	1280

Note that the dbspace can be decreased in 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.

For a description of the full syntax and actions of the ALTER DBSPACE command, see Chapter 6, “SQL Statements” in the *Sybase IQ Reference Manual*.

Changing the mode of a dbspace

The read-write mode of a dbspace controls whether writes to the dbspace are performed and if data should be relocated from the dbspace. A dbspace has one of three read-write modes, which is changed using the ALTER DBSPACE command:

- **Read-write (RW):** allocations can be made from the dbspace and writes are allowed (the default for a newly created dbspace)
- **Read-only (RO):** no writes are performed for main or local dbspaces, but modifications to existing objects are allowed; new versions are instead written to other read-write dbspaces
- **Relocate (RR):** allocations from the dbspace are not allowed and objects are subject to relocation to read-write dbspaces; main relocate dbspaces are also read-only

The system stored procedure sp\_iqdbspace lists the read-write mode of one or all dbspaces in the RWMode column.

Set the read-write mode of a dbspace to read-only (RO) to prevent further writes to the dbspace. The following ALTER DBSPACE statement changes the read-write mode of the dbspace dbspacedb2 to read-only. The INSERT statement allocates storage for the new version from the remaining read-write (RW) dbspaces.

```
ALTER DBSPACE dbspacedb2 READONLY;
INSERT t1 VALUES (2);
sp_iqdbspace;
```

In the sp\_iqdbspace output, note that the dbspace dbspacedb2 now has a read-write mode of read-only (RO in the RWMode column).

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ__ SYSTEM_ MAIN	D:\IQ\br/>dbspacedb.iq	MAIN	RW	20	15M	95M	8K	1H,64F, 33R,32D, 78A,37O, 128M	1	1920
dbspacedb2	D:\IQ\br/>dbspacedb.iq2	MAIN	RO	5	10M	20M	8K	1H,32F, 37A,19O	1045440	1046719
dbspacedb3	D:\IQ\br/>dbspacedb.iq3	MAIN	RW	11	10M	40M	8K	1H,32F, 62A,38O	2090880	2092159
IQ__ SYSTEM_ TEMP	dbspacedb. iqtmp	TEMPO RARY	RW	9	10M	10M	8K	1H,64F, 11A,37O	1	1280

**Relocating objects and data**

Tables and user data in a dbspace must be dropped or relocated to other dbspaces before the dbspace can be dropped. The two step relocation process in this example first uses the ALTER DBSPACE command to change the mode of the IQ\_SYSTEM\_MAIN dbspace to relocate mode, so that no further allocations are made from the dbspace and objects can be relocated:

```
ALTER DBSPACE IQ_SYSTEM_MAIN relocate;
```

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_MAIN	D:\IQ\ dbspacedb.iq	MAIN	RR	4	15M	95M	8K	1H,64F, 62A	1	1920
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RO	5	10M	20M	8K	1H,32F, 56A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	25	10M	40M	8K	1H,64F 33R,32D ,59A, 128M	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 32A	1	1280

Note that the read-write mode of the IQ\_SYSTEM\_MAIN dbspace is now RR for relocate. Setting the dbspace read-write mode to relocate also has the effect that no future writes to the dbspace will be performed.

The second step in the process is relocating objects and data in the relocate dbspace. Individual objects in the dbspace are relocated using the sp\_iqrelocate system stored procedure. For example, the following command relocates the index t1c1hg on table t1 and relocates the entire table t2:

```
sp_iqrelocate 'index t1c1hg table t2';
```

ObjectName	NRelocated	RelocStatus
t1.DBA.t1c1hg	19	relocated
t2	4	relocated
t2.DBA.ASIQ_IDX_T430_C1_FP	17	relocated
t2.DBA.t2c1hng	0	no relocs

The `sp_iqrelocate` stored procedure works much like the `UPDATE` statement. Queries can continue to access tables while `sp_iqrelocate` is running. Data manipulation (DML) and data definition (DDL) commands may return errors, since the tables are locked for update. Likewise, `sp_iqrelocate` may return an error if another connection is updating a table. To avoid these write access conflict errors, you can change object-level permission to ensure that there is no write activity on them or you can limit connections with single-user mode.

All data on dbspaces with the read-write mode of relocate can be relocated using a single `sp_iqrelocate` command. The following command relocates all data on relocate dbspaces in the `empdb` database, which in this case is all data in the `IQ_SYSTEM_MAIN` dbspace:

```
sp_iqrelocate 'empdb' ;
```

ObjectName	NRelocated	RelocStatus
t1	5	relocated
t1.DBA.ASIQ_IDX_T429_C1_FP	17	relocated
t1.DBA.t1c1hg	0	no relocs
t2	0	no relocs
t2.DBA.ASIQ_IDX_T430_C1_FP	0	no relocs
t2.DBA.t2c1hng	0	no relocs

Note that the four objects with relocation status of no relocs were relocated by the earlier `sp_iqrelocate` command.

The column `NRelocated` in the `sp_iqrelocate` output is the number of blocks that were relocated for each object. In addition to relocated (all blocks in relocate dbspaces were relocated), other possible values of the `RelocStatus` column are partial (the number of blocks in relocate dbspaces exceeds the `sp_iqrelocate maxsize` parameter) and no relocs (the object has no blocks in relocate dbspaces).

`sp_iqrelocate` does not automatically commit. You must commit the changes before they are persistent.

For a full description of the syntax, actions, and output of `sp_iqrelocate`, see Chapter 10, “System Procedures” in the *Sybase IQ Reference Manual*.

Dropping a main  
dbspace

After all objects and user data are relocated to other dbspaces, the dbspace is ready to drop using the `DROP DBSPACE` command. Before executing the `DROP DBSPACE` command, check the block type information in the `sp_iqdbspace` output to be sure the dbspace does not have blocks of type ‘A’ or ‘O’ (blocks with data from active table versions or old versions that may still be in use).

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_MAIN	D:\IQ\ dbspacedb.iq	MAIN	RR	0	15M	95M	8K	1H,64F	1	1920
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RO	2	10M	20M	8K	1H,32F, 21A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	34	10M	40M	8K	1H,64F, 33R,32D ,156A,18 X,128M	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqttmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 32A	1	1280

The following command drops the dbspace IQ\_SYSTEM\_MAIN, from which all data was relocated:

```
DROP DBSPACE IQ_SYSTEM_MAIN;
```

The `sp_iqdbspace` output shows that the dbspace IQ\_SYSTEM\_MAIN has been dropped:

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RO	2	10M	20M	8K	1H,32F, 21A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	33	10M	40M	8K	1H,64F, 33R,32D ,156A, 128M	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqttmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F, 32A	1	1280

### Reusing space

The blocks formerly used by the IQ\_SYSTEM\_MAIN dbspace can now be used by a new dbspace, if the new dbspace is not larger than the old dbspace that was dropped. The following CREATE DBSPACE command creates the new 10MB dbspace dbspacedb1, which is the same size as the original IQ\_SYSTEM\_MAIN dbspace:

```
create dbspace dbspacedb1 as
    'D:\IQ\dbspacedb.iq1'
    size 10 reserve 20;
```

Note in the sp\_iqdbspace output that the new dbspace dbspacedb1 uses the same blocks that were made available by dropping IQ\_SYSTEM\_MAIN (see the columns FirstBlk and LastBlk).

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb1	D:\IQ\	MAIN	RW	4	10M	20M	8K	1H,32F, 17R	1	1280
dbspacedb2	dbspacedb.iq1									
dbspacedb2	D:\IQ\	MAIN	RO	2	10M	20M	8K	1H,32F ,21A	1045440	1046719
dbspacedb3	dbspacedb.iq2									
dbspacedb3	D:\IQ\	MAIN	RW	32	10M	40M	8K	1H,64F, 16R,32D, 156A, 128M	2090880	2092159
dbspacedb3	dbspacedb.iq3									
IQ_SYSTEM_TEMP	dbspacedb.	TEMPO	RW	8	10M	10M	8K	1H,64F, 32A	1	1280
	iqttmp		RARY							

### Dropping a temporary dbspace

To drop a temporary dbspace, use the same commands and stored procedures (except sp\_iqrelocate) used to drop a main dbspace. This example drops the temporary dbspace IQ\_SYSTEM\_TEMP. First, create a second temporary dbspace dbspacedbttmp2 to replace the dbspace that you are dropping.

```
create dbspace dbspacedbttmp2 as
    'D:\IQ\dbspacedb.iqttmp2'
    iq temporary store size 20;
```

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb1	D:\IQ\	MAIN	RW	6	10M	20M	8K	1H,32F,33R	1	1280
dbspacedb1	dbspacedb.iq1									

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RO	2	10M	20M	8K	1H,32F,21A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	30	10M	40M	8K	1H,64F, 32D,156A, 128M	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RW	8	10M	10M	8K	1H,64F,32A	1	1280
dbspacedbtmp2	D:\IQ\ dbspacedb.iqtmp2	TEMPO RARY	RW	2	20M	0B	8K	1H,32F	1045440	1047999

Use the ALTER DBSPACE command to change the read-write mode of the IQ\_SYSTEM\_TEMP dbspace to relocate (RR), so that no further allocations are made from the dbspace and objects can be relocated:

```
alter dbspace IQ_SYSTEM_TEMP relocate;
sp_iqdbspace IQ_SYSTEM_TEMP;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RR	7	10M	10M	8K	1H,64F,16A	1	1280

Temp space may be in use by other connections. Run the sp\_iqconnection stored procedure to find temporary space that is in use. The columns TempTableSpaceKB and TempWorkSpaceKB indicate that temp space is being used by that connection. You may need to rollback or disconnect these connections to release this space.

```
sp_iqconnection;
```

ConnHandle	Name	Userid	...	TempTableSpaceKB	TempWorkSpaceKB	...
2		DBA	...	0	128	...

You may also need to issue a CHECKPOINT to release remaining Temp Store blocks:

```
checkpoint;
```

Run the sp\_iqdbspace stored procedure to see if the IQ\_SYSTEM\_TEMP dbspace is empty. The temporary dbspace can be dropped if it contains only block types 'F' and 'H' (free list blocks and header blocks of the free list).

```
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb1	D:\IQ\ dbspacedb.iq1	MAIN	RW	6	10M	20M	8K	1H,32F, 33R	1	1280

## Dbspace management example

---

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RO	2	10M	20M	8K	1H,32F,21A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	30	10M	40M	8K	1H,64F,32D,156A,128M	2090880	2092159
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RR	6	10M	10M	8K	1H,64F	1	1280
dbspacedbttmp2	D:\IQ\ dbspacedb.iqtmp2	TEMPO RARY	RW	4	20M	0B	8K	1H,64F,16A	1045440	1047999

You can also run `sp_iqdbspace` with the dbspace name to see if the `IQ_SYSTEM_TEMP` dbspace is empty:

```
sp_iqdbspace IQ_SYSTEM_TEMP;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
IQ_SYSTEM_TEMP	dbspacedb.iqtmp	TEMPO RARY	RR	6	10M	10M	8K	1H,64F	1	1280

When the original temporary dbspace is empty, run `DROP DBSPACE` to drop `IQ_SYSTEM_TEMP`:

```
drop dbspace IQ_SYSTEM_TEMP;
sp_iqdbspace;
```

Name	Path	Segment Type	RW Mode	Usage	DBS Size	Reserve	Stripe Size	Blk Types	First Blk	Last Blk
dbspacedb1	D:\IQ\ dbspacedb.iq1	MAIN	RW	6	10M	20M	8K	1H,32F,33R	1	1280
dbspacedb2	D:\IQ\ dbspacedb.iq2	MAIN	RO	2	10M	20M	8K	1H,32F,21A	1045440	1046719
dbspacedb3	D:\IQ\ dbspacedb.iq3	MAIN	RW	30	10M	40M	8K	1H,64F,32D,156A,128M	2090880	2092159
dbspacedbttmp2	D:\IQ\ dbspacedb.iqtmp2	TEMPO RARY	RW	4	20M	0B	8K	1H,64F,16A	1045440	1047999

The original temporary dbspace `IQ_SYSTEM_TEMP` is gone and the new temporary dbspace `dbspacedbttmp2` is now used as the temporary dbspace for the database `dbspacedb`.

## Working with tables

When you create a database, the only tables in it are the **system tables**, which hold the database schema.

This section describes how to create, alter, and delete tables from a database. The examples can be executed in DBISQL, but the SQL statements are independent of the administration tool you are using.

### Creating tables in multiplex databases

For multiplex databases, all schema changes to be shared must be done on the write server. Follow these steps:

- On the write server, execute DDL (Database Definition Language) commands.
- Create database objects interactively or with scripts, just as you would with any Sybase IQ database. For complete syntax, see the *Sybase IQ Reference Manual*.
- Insert the data. Perform the inserts on the write server while the multiplex runs.

---

**Note** If you use Interactive SQL to load tables, be sure to enter a COMMIT command.

---

You may want to 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.

## Creating tables

### Creating tables in Sybase Central

To create a table using Sybase Central, see “Managing tables” in *Introduction to Sybase IQ*.

### SQL statement for creating tables

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 sample database with user ID DBA and password SQL.

For information on connecting to the sample database from DBISQL, see “Connecting to the sample database from Sybase Central or DBISQL.”

## Creating tables with SQL

You can create tables with other tools in addition to DBISQL. The SQL statements described here are independent of the tool you are using.

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. To create a primary key, see “Creating primary and foreign keys” on page 247.

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 “CREATE TABLE statement” in the *Sybase IQ Reference Manual*. For information about building constraints into table definitions using CREATE TABLE, see Chapter 9, “Ensuring Data Integrity”.

## 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. In the *Sybase IQ Reference Manual*, see Chapter 4, “SQL Data Types,” for a list of supported data types, or see the CREATE DOMAIN statement for details on how to create a customized data type.

## 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.

Populating persistent tables in multiplex databases

Certain client tools need to create and populate persistent tables. In a multiplex database, persistent tables have two locations, depending upon the server to which the user connects:

- On the write server, users create persistent tables in the shared IQ Main Store.
- On the query server, users create persistent tables in a local IQ Main Store (Local Store). A dbspace must be created to serve as the local IQ Main Store before users create any persistent objects. If no local IQ Main Store exists, creating persistent objects returns an error.

---

**Note** While executing DDL statements propagated from the write server, Sybase IQ resolves conflicts by renaming query server persistent objects that conflict. See “Resolving static collisions” on page 212.

---

**Table 5-9: Persistent objects in multiplex databases**

Write server	Query server
Persistent tables reside in the shared IQ Main Store and are visible to all servers in the multiplex. The write server performs all changes to these objects (DML and DDL)	Persistent tables reside in the local IQ Main Store. They are local to a particular query server in the multiplex, and are visible to all users of that query server. The persistent tables are not visible to users of other query servers in the multiplex.
DDL executed on write server persistent objects automatically propagates to query servers.	DML and DDL executed on query server persistent objects affect only the local IQ Main Store.
Creating temporary tables	<p>There are two types of temporary tables, global and local.</p> <p>You <i>create</i> 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.</p>

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 the write server are also created on all query servers. Global temporary tables created on the query server are local to that query server.

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

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

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.

See “Versioning of temporary tables” for versioning information on local temporary tables.

#### 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.

#### 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 5-10: Table placement**

Type of table	Permitted placement	Default placement
Permanent	IQ Store, Catalog Store, IQ Local Store (query server only)	IQ Store, IQ Local Store (query server only)
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. Run these commands from the write server after starting it in single-node mode, and then synchronize query servers. For an example of starting in single-node mode, see “Replacing a write server with intact files” on page 667.

**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.

For complete information on IQ indexing, see Chapter 6, “Using Sybase IQ Indexes”

**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 the following table for implications of IQ UNIQUE.

**Table 5-11: Effect of IQ UNIQUE**

IQ UNIQUE 256 or less	IQ UNIQUE 65536 or less	IQ UNIQUE unspecified or greater than 65536
Storage optimized for small number of unique values	Storage optimized for medium number of unique values	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	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.	No extra cache needed for loads
Loads may be slower if you have numerous columns with IQ UNIQUE <256	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 “MINIMIZE\_STORAGE option” in *Sybase IQ Reference Manual*.

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 Chapter 10, “System Procedures” in *Sybase IQ Reference Manual*.

#### 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.

## 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 of the ALTER TABLE command, see *Sybase IQ Reference Manual*. For information about building constraints into table definitions using ALTER TABLE, see Chapter 9, “Ensuring Data Integrity”

Altering tables in Sybase Central

The property sheets for tables and columns display all the table or column attributes. You can alter a table definition in Sybase Central by displaying the property sheet for the table or column you wish to change, altering the property, and clicking OK to commit the change.

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.

## 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 are very fast, taking only seconds to occur. The size of the data does not effect the speed of the operation.

For a full description of the DROP statement, see *Sybase IQ Reference Manual*.

❖ **Dropping a table in Sybase Central**

- 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 and foreign keys

The CREATE TABLE and ALTER TABLE statements allow many attributes of tables to be set, including column constraints and checks. This section shows how to set table attributes using the primary and foreign keys as an example.

### 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. Each row in a table is uniquely identified by its primary key.

Columns in the primary key are not allowed to contain NULL. You must specify NOT NULL on the column in the primary key.

### Creating a primary key in Sybase Central

❖ **Creating a primary key in Sybase Central**

- 1 Connect to the database.
- 2 Click the Tables folder for that database.
- 3 Right-click the table you wish to modify, and select Properties from the pop-up menu to display its property sheet.
- 4 Click the Columns tab, select the column name, and either click Add to Key or Remove from Key. *Column values must be unique.*

For more information, see the Sybase Central online Help.

---

**Note** Primary key column order is based on the order of the columns during table creation. It is not based on the order of the columns as specified in the primary key declaration.

---

## Creating foreign keys

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 skill 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 the chapter “SQL Language Elements” in the *Sybase IQ Reference Manual*.

## Creating a foreign key in Sybase Central

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.

### ❖ Creating a foreign key in Sybase Central

- 1 Connect to the database.
- 2 Click the Tables folder for that database.
- 3 Click the table holding the primary key, and drag it to the foreign key table.

- 4 When the primary key table is dropped on the foreign key table, the Foreign Key Wizard is displayed, which leads you through the process of creating the foreign key.

For more information, see the Sybase Central online Help.

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

## Table information in the system tables

All the information about tables in a database is held in the system tables. The information is distributed among several tables. For more information, see Chapter 9, “System Tables,” in *Sybase IQ Reference Manual*.

You can use Sybase Central or DBISQL to browse the information in these tables. Type the following command in the DBISQL command window to see all the columns in the SYS.SYSTABLE table:

```
SELECT *
FROM SYS.SYSTABLE
```

### ❖ Displaying system tables in Sybase Central

- 1 Connect to the database.
- 2 Right-click the database, and select Filter Objects from the pop-up menu.
- 3 Select SYS and OK.
- 4 When you view the database tables or views with Show System Objects checked, the system tables or views are also shown.

## Working with views

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.

Differences between views and permanent tables

- You can perform INSERT and DELETE operations on some views.
- You can create views based on other views.

Benefits of tailoring access

There are some differences between views and permanent tables:

- You cannot create indexes on views.
- You cannot perform INSERT, DELETE, and UPDATE operations on all 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.

Views are used to tailor access to data in the database. Tailoring access serves several purposes:

- **Improved security** By not allowing access to information that is not relevant.
- **Improved usability** By presenting users and application developers with data in a more easily understood form than in the base tables.
- **Improved consistency** By centralizing in the database the definition of common queries.

## Creating views

A SELECT statement operates on one or more tables and produces a result set that is also a table: just 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 dept_ID, count(*)  
FROM employee  
GROUP BY dept_ID
```

You can create a view containing the results of this statement as follows:

```
CREATE VIEW DepartmentSize AS  
SELECT dept_ID, count(*)  
FROM employee  
GROUP BY dept_ID
```

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 employee 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.

❖ **Creating a view in Sybase Central**

- 1 Connect to the database.
- 2 Click the Views folder for that database.
- 3 Double-click Add View.
- 4 Enter the tables and columns to be used. For instance, to create the same view as in the SQL example shown above, enter employee and dept\_ID.
- 5 From the File menu select Execute Script and from the File menu select Close.

## 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
```

Inserting and deleting  
on 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.

---

## 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
    (Dept_ID, NumEmployees)
AS
    SELECT dept_ID, count(*)
    FROM Employee
    GROUP BY dept_ID
```

## Permissions on views

A user may perform an operation through a view if one or more of the following are true:

- The appropriate permission(s) on the view for the operation has been granted to the user by a DBA.
- The user has the appropriate permission(s) on all the base table(s) for the operation.

- The user was granted appropriate permission(s) 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.

## Deleting 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
```

Dropping 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.

## Views in the system tables

All the information about views in a database is held in the system table SYS.SYSTABLE. The information is presented in a more readable format in the system view SYS.SYSVIEWS. For more information about these, see *Sybase IQ Reference Manual*.

You can use DBISQL to browse the information in these tables. Type the following statement in the DBISQL command window to see all the columns in the SYS.SYSVIEWS view:

```
SELECT *
  FROM SYS.SYSVIEWS
```

To extract a text file containing the definition of a specific view, use a statement such as the following:

```
SELECT viewtext FROM SYS.SYSVIEWS
  WHERE viewname = 'DepartmentSize';
  OUTPUT TO viewtext.sql
  FORMAT ASCII
```

## **Working with indexes**

Performance is a vital consideration when designing and creating your database. Sybase IQ indexes dramatically improve the performance of database searches over searches in traditional relational databases. 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

You can create column indexes in three ways:

- With the CREATE INDEX command
- 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 New Join Index wizard in Sybase Central

See Chapter 6, “Using Sybase IQ Indexes” for details on selecting and creating indexes. See the *Sybase IQ Reference Manual* for command syntax. See Chapter 5, “Indexing and Loading Data” in the *Introduction to Sybase IQ* for Sybase Central instructions.

## Indexes in the system tables

Information on indexes is in the system tables SYSINDEX, SYSIQINDEX, SYSIXCOL, and for join indexes, SYSIQJINDEX. See the *Sybase IQ Reference Manual* for a description of these tables.

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 department table, issue the command:

```
sp_iqindex 'department'
```

The following information displays.

table_name	table_owner	column_name	index_type	index_name	unique_index	remarks
department	DBA	dept_head_id	FP	ASIQ_IDX_T201_C3_FP	N	(NULL)
department	DBA	dept_id	FP	ASIQ_IDX_T201_C1_FP	N	(NULL)
department	DBA	dept_id	HG	ASIQ_IDX_T201_C1_HG	U	(NULL)
department	DBA	dept_name	FP	ASIQ_IDX_T201_C2_FP	N	(NULL)

If you omit the table name from the command, `sp_iqindex` displays this information for all tables in the database.

## Validating indexes

You can validate an index on Adaptive Server 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 complements the validity checking carried out by the Adaptive Server 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.

```
dbvalid -i EmployeeIndex
```

For more information, see *Sybase IQ Utility Guide*.

## Renaming indexes

You can rename an index in a base table or global temporary table with the owner type `USER`. See the `ALTER INDEX` command in the *Sybase IQ Reference Manual* for more information on renaming indexes and changing foreign key role names. Note that indexes created to enforce key constraints cannot be renamed.

## Removing 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. Before you drop a join index, see “Modifying tables included in a join index” for special requirements.

# Using Sybase IQ Indexes

## About this chapter

This chapter describes Sybase IQ index 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.

## Contents

Topic	Page
Overview of indexes	257
Creating Sybase IQ indexes	261
Choosing an index type	264
Sybase IQ index types	270
Adding column indexes after inserting data	286
Using join indexes	287
Estimating the size and benefit of a join index	307

## Overview of indexes

Indexes are used to improve data retrieval performance. Traditional indexes 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 processing 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

When you load data into a table, Sybase IQ 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. Column storage structures your data according to the attributes you are interested in tracking. In a data warehousing environment, usually you want to look at specific attributes of thousands or millions of rows of data, rather than complete, single rows of data that typically are 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. 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 this default index with one or more of the Sybase IQ bitwise index types. You can choose from several column index types. The column indexes you define are created as part of each individual table.

Besides the column indexes, Sybase IQ also allows you to define *join indexes*. Join indexes are optimized for joining related tables. You may want to create a join index for any set of columns that your users commonly join to resolve queries. Column indexes underlie any join indexes involving those columns.

The first half of this chapter discusses column indexes. The second half of this chapter discusses join indexes; see “Using join indexes” for details.

A **default index** that optimizes projections is created by Sybase IQ for all columns.

Columns with fewer than 65536 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 for most columns that best represent the cardinality and usage of column data:

- 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 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 DSS operations involving ranges or aggregates
- Low\_Fast or LF A value-based bitmap for processing queries on low-cardinality data (recommended for up to 1,000 distinct values. 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 or VARCHAR column as a delimited list.

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.

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 to the left or right of the decimal point. Be aware that some index types are incompatible, and that creating indexes you don't need wastes a lot of disk space. Read the sections that follow for details on how to select an index.

When a column is designated as a 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 CREATE INDEX statement in *Sybase IQ Reference Manual*.

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 type(s).
Index guidance from the optimizer	If multiple indexes are defined on a particular column, Sybase IQ builds all the indexes for that column from the same input data.  If you set the INDEX_ADVISOR option on your database, Sybase IQ issues messages in the message log or query plan to suggest additional index(es) that might improve performance. Messages focus on these areas:
Adding and dropping indexes	<ul style="list-style-type: none"><li>• local predicate columns</li><li>• single-column join key columns</li><li>• correlated subquery columns</li><li>• grouping columns</li></ul> For details, see “INDEX_ADVISOR option” in Chapter 2, “Database Options,” of <i>Sybase IQ Reference Manual</i> .  If you decide to follow the recommendations, you must create the indexes yourself.  If you discover later that an additional index is needed, you can always add indexes. However, it is much faster to create all the appropriate 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 the <i>Sybase IQ Reference Manual</i> 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 *Sybase IQ Reference Manual* 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 these benefits over traditional indexing techniques:

- Index sizes 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. Most traditional databases need three times as much 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.

## Creating Sybase IQ indexes

You can create a column index explicitly using either the CREATE INDEX statement or Sybase Central. These two methods are discussed in the sections that follow.

## The CREATE INDEX statement

To create an Sybase IQ column index, use this syntax:

```
CREATE [ UNIQUE ] [ index-type ] INDEX index-name
... ON [ owner.]table-name
... ( column-name [, column-name]...)
... [ { IN | ON } dbspace-name ]
... [ NOTIFY integer ]
... [ DELIMITED BY 'separators-string' ]
... [ LIMIT maxwordsize-integer ]
```

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.

### Examples

The first example creates a High\_Non\_Group (HNG) index called ship\_ix on the ship\_date column of the sales\_order\_items table.

```
CREATE HNG INDEX ship_ix
    ON dbo.sales_order_items (ship_date)
```

The second example creates a Low\_Fast index called sales\_order\_region on the region column of the sales\_order table.

```
CREATE LF INDEX sales_order_region
    ON dbo.sales_order (region)
```

For examples of how to create a CMP index, see “The Compare (CMP) index type” on page 276.

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. To prevent these messages, specify NOTIFY 0.

You can 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 Adaptive Server Anywhere tables. Note that, if one of these CREATE INDEX statements fails, all of them roll back. For more information, see the *Sybase IQ Reference Manual*.

---

**Note** You cannot place an index in a particular dbspace. Sybase IQ always places an index in the same type of dbspace (IQ Store or Temporary Store) as its table. When you load the index, the data is spread across any database files of that type with room available. The *dbspace-name* option of CREATE INDEX is ignored for IQ indexes, and is provided for compatibility with Adaptive Server Anywhere.

---

## Creating an index with Sybase Central

To create a column index using Sybase Central, follow these steps.

❖ **Creating an index with Sybase Central**

- 1 Connect to the database.
- 2 Select the table in which the column appears.
- 3 Open the Indexes folder.
- 4 Double-click the Add Index icon, enter a name for the index, and click Next.
- 5 Select an index type. A High Group is created if you do not click another index type.
- 6 Optionally set the number of records added before each notification message, and click Next.
- 7 Select the column(s) you want to index, and click Next.
- 8 Enter attributes for the index as appropriate, and click Finish to create the index.

## Creating indexes concurrently

In some cases, you can create more than one column index at the same time:

- Each CREATE INDEX statement can create only one index.
- Each connection can create only one index at a time.
- 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.

## **Choosing an index type**

The set of indexes 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 this number reaches certain levels, choose indexes according to the recommendations in Table 6-1.

**Table 6-1: 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

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 6-2: 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	LF, HG
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	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

---

**Note** 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, *every* column named in the WHERE and GROUP BY clause 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 and CMP index types; CHAR and VARCHAR data greater than 255 bytes can only be used in the default, CMP, 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.

## Indexing criteria: 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 6-3: 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.

## Data types in the index

The default index allows any data type. See the following table for a list of other indexes supported for each data type.

**Table 6-4: Indexes supported for data types**

Data type	Supported indexes	Unsupported indexes
tinyint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
smallint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
int	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
unsigned int	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
bigint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
unsigned bigint	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
numeric, decimal	CMP, HG, HNG, LF	WD, DATE, TIME, DTTM
double	LF (HG permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM
float	LF (HG permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM
real	LF (HG permitted but not recommended)	CMP, HNG, WD, DATE, TIME, DTTM
bit	(Default index only)	CMP, HG, HNG, LF, WD, DATE, TIME, DTTM
date	CMP, HG, HNG, LF, DATE	WD, TIME, DTTM
time	CMP, HG, HNG, LF, TIME	WD, DATE, DTTM
datetime, timestamp	CMP, HG, HNG, LF, DTTM	WD, DATE, TIME
char <= 255 bytes, character	CMP, HG, HNG, LF, WD	DATE, TIME, DTTM
char >255 bytes	CMP, WD	HG, HNG, LF, DATE, TIME, DTTM
varchar <= 255 bytes	CMP, HG, HNG, LF, WD	DATE, TIME, DTTM
varchar >255 bytes	CMP, WD	HG, HNG, LF, DATE, TIME, DTTM
binary	CMP, HG, LF	HNG, WD, DATE, TIME, DTTM
varbinary <= 255 bytes	CMP, HG, LF	HNG, WD, DATE, TIME, DTTM
varbinary > 255 bytes	CMP	HG, HNG, LF, WD, DATE, TIME, DTTM

## Combining index types

If a column is going to be used in more than one type of query, more than one column index type might be appropriate. Table 6-5 shows which index types make good combinations.

**Table 6-5: Mix of valid index**

Existing Index	Add Index					
	HG	HNG	LF	CMP <sup>a</sup>	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 = An unlikely combination

a. A CMP index applies to a pair of columns. Each of those columns always has at least one other index.

---

**Note** The High\_Group index in Sybase IQ Version 12 differs from earlier versions. For some columns you may want both High\_Group and High\_Non\_Group; previously, it did not make sense to have both. 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.

### Default column index

For any column that has no index defined, or whenever it is the most effective, query results are produced using the default index. This structure is fastest for projections, but generally is slower than any of the three column index types you define for anything other than a projection. Performance is still faster than most RDBMSs since one column of data is fetched, while other RDBMSs need to fetch all columns which results in more disk I/O operations.

Sybase IQ supports the lookup of default indexes on fixed-width columns that are one or two bytes wide.

The `sp_iqindexmetadata` stored procedure generates a report describing a specified index or indexes belonging to a specified owner or table. For example, the output allows easy checking of whether a given index is a 1-byte or 2-byte default index.

Sybase recommends that if users see “Old One Byte FP” (or “Old Two Byte FP”), they rebuild the default index to leverage the newer, faster index structure. After the rebuild, users see the typical “One Byte FP”.

For details, see “`sp_iqindexmetadata` procedure” in the *Sybase IQ Reference Manual*.

## Projections on few rows

If a column is used only in projections, even if some of the queries return a small number of rows, Low\_Fast and High\_Non\_Group indexes are redundant because the default structure is equally as fast for projecting a small number of rows.

## 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

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 6-6: LF advantages/disadvantages**

<b>Advantages</b>	<b>Disadvantages</b>
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

**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

Use an HG index when:

- The column will be used in a join predicate
- A column has more than 1000 unique values

---

**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

The following table lists advantages and disadvantages of High\_Group indexes.

**Table 6-7: HG advantages/disadvantages**

<b>Advantages</b>	<b>Disadvantages</b>
<p>Quickly processes queries with GROUP BY.</p> <p>This index facilitates join index processing. It is one of indexes recommended for columns used in join relationships. LF is the other.</p>	<p>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).</p> <p>This index type takes the longest time to populate with data, and to delete. To optimize its delete performance, see “Optimizing delete operations” in <i>Sybase IQ Performance and Tuning Guide</i>.</p> <p>Cannot use this index if data in your columns is BIT, VARBINARY &gt; 255 bytes, CHAR &gt; 255 bytes, or VARCHAR &gt; 255 bytes.</p> <p>This index is not recommended for FLOAT, REAL, and DOUBLE data.</p>

## Comparison with 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 the “Notes” section of the CREATE INDEX statement in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

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

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 6-8: HNG advantages/disadvantages**

Advantages	Disadvantages
<p>Due to compression algorithms used, disk space requirements can be reduced without sacrificing performance.</p> <p>If the column has a high number of unique values, this is the fastest index, with few exceptions described below.</p>	<p>This index is not recommended for GROUP BY queries.</p> <p>Index not possible if uniqueness enforced.</p> <p>Cannot use this index if data in your columns is FLOAT, REAL, DOUBLE, BIT, BINARY, VARBINARY, CHAR &gt; 255 bytes, or VARCHAR &gt; 255 bytes.</p>

## Comparison to other 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.

## Recommended use

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.
- Even though a CMP index can be created on columns defined as NULL, a partial width insert into a table is disallowed when not all columns of a CMP index are part of the insert.
- 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

This index allows you to store words from a column string of CHAR and VARCHAR data.

### 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 or VARCHAR column in a base table
- The minimum permitted column width is 3 bytes and the maximum permitted column width is 32767 bytes.
- You must enclose the list of delimiters in single quotes. The Sybase Central Add 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 will be 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 company_name
contains ('Farms')
```

The parser determines that the string contains

'Farms'

instead of

'Farms'

and returns 0 instead of 1. You can avoid this problem by using VARCHAR instead of CHAR columns.

- This index cannot be repaired with the consistency checker.

## Advantages and disadvantages of WD

See the following table for advantages and disadvantages of using a WD index.

**Table 6-9: WD advantages/disadvantages**

Advantages	Disadvantages
<p>Huge performance gains are possible for large loads.</p> <p>Certain LIKE predicates execute faster with this index.</p> <p>CONTAINS predicate used with this index takes precedence over the LIKE predicate.</p> <p>Best way to index keywords or parts of a URL.</p>	<p>Disk space requirements may potentially be very large.</p> <p>Index not possible if uniqueness enforced.</p> <p>Can only use this index if data in your columns is CHAR or VARCHAR.</p>

## 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

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. See also the section “Additional indexes” on page 284 for index recommendations for DATE, TIME, DATETIME, and TIMESTAMP columns.

---

The table tab used in the examples in this section contains columns defined as follows:

```
CREATE TABLE tab
( col1 DATE,
  col2 DATETIME,
  col3 TIME);
```

### Queries with DATEPART equality, range, and IN list predicates

For a query with an equality predicate (= or !=), if one side of the comparison is a DATEPART expression or some other date and time function (for example, YEAR, QUARTER, DAY, MINUTE), and the other side of the comparison is a constant expression (including a constant value or host variable), then the DATE, TIME, or DTTM index is used (if the index is available) to get the result set.

For example, the DATE, TIME, or DTTM index is used in 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

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 6-10: DATE/TIME/DTTM advantages/disadvantages**

<b>Advantages</b>	<b>Disadvantages</b>
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

The recommendation is 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).

## Optimizing performance for ad hoc joins

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

Here is a quick chart that summarizes how to select an index type.

<b>Criteria to identify</b>	<b>Index to select</b>
Note indexes created automatically on all columns.	<i>Default index</i>
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

<b>Criteria to identify</b>	<b>Index to select</b>
Identify columns that contain a low number of unique values and do not already use multiple indexes.	LF
Identify columns that have a high number of unique values and that are part of a GROUP BY clause in a select list in a SELECT DISTINCT or DISTINCT COUNT.	HG
Identify columns that may be used in the WHERE clause of ad hoc join queries that do not already have HG or LF indexes.	HG or LF
Identify columns that have a high number of unique values and that will not be used with GROUP BY, SELECT DISTINCT or DISTINCT COUNT.	HNG
Identify pairs of columns with the same data type, precision, and scale that are likely to need frequent comparison.	CMP
Identify columns that contain a list of keywords or a URL.	WD
Identify columns of DATE, TIME, DATETIME, or TIMESTAMP that have a high number of unique values and that will <i>not</i> 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. For information on synchronization, see “Synchronizing join indexes”.

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.

---

## 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.

### 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 “Join hierarchy overview” below for details.

## 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.

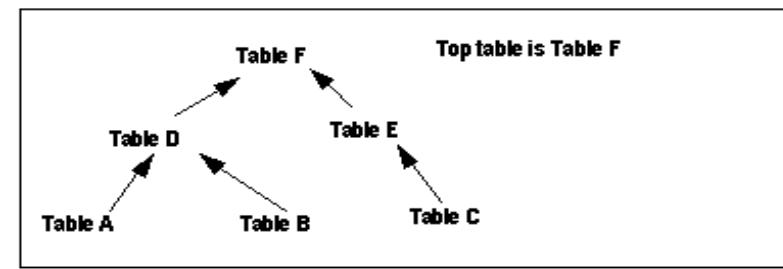
## Join hierarchy overview

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 6-1: 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

## 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. For examples of these messages, see “Interpreting notification messages” on page 343.

Join index columns must have identical data type, precision, and scale.

## 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 Figure 6-1 on page 289. 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.

## 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. For more information, see the *Sybase IQ Performance and Tuning Guide*.

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 asiqdemo database, the id in the customer table and the cust\_id in the sales\_order table each contain a customer ID. The customer table contains one row for each customer and, therefore, has a unique value in the id column in each row. The sales\_order table contains one row for each transaction a customer has made. Presumably, there are many transactions for each customer, so there are multiple rows in the sales\_order table with the same value in the cust\_id column.

So, if you join customer.id to sales\_order.cust\_id, the join relationship is one-to-many. As you can see in the following example, for every row in customer, there are potentially many matching rows in sales\_order.

```
select sales_order.id, sales_order.cust_id,
       customer.lname
  from sales_order, customer
 where sales_order.cust_id = customer

id cust_id id lname
2583,101,101,'Devlin'
2001,101,101,'Devlin'
2005,101,101,'Devlin'
2125,101,101,'Devlin'
2206,101,101,'Devlin'
2279,101,101,'Devlin'
2295,101,101,'Devlin'
2002,102,102,'Reiser'
2142,102,102,'Reiser'
2318,102,102,'Reiser'
2338,102,102,'Reiser'
2449,102,102,'Reiser'
2562,102,102,'Reiser'
2585,102,102,'Reiser'
2340,103,103,'Niedringhaus'
2451,103,103,'Niedringhaus'
2564,103,103,'Niedringhaus'
2587,103,103,'Niedringhaus'
2003,103,103,'Niedringhaus'
2178,103,103,'Niedringhaus'
2207,103,103,'Niedringhaus'
```

2307,103,103,'Niedringhaus'

---

**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.

Each column that is a foreign key, or a component of a foreign key, needs its own HG index. For multicolumn primary keys, if any column or subset of columns is used as a foreign key, you need to create an HG index explicitly on each such column. The HG index created automatically for the multicolumn primary key does not suffice for this purpose.

If your queries include joins on the multicolumn primary key, then you should define the multicolumn index, even if you are not using referential integrity. Creating the multicolumn index can help optimize these queries.

- 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 New Join Index Wizard. You specify the join hierarchy as part of this step, as described in “Join hierarchy overview”.
- 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.

## 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 details on inserting and deleting data, see Chapter 7, “Moving Data In and Out of Databases.” For complete syntax of the CREATE TABLE, ALTER TABLE, LOAD TABLE, INSERT INTO, and SYNCHRONIZE commands, see the *Sybase IQ Reference Manual*. The sections that follow give details on other steps in creating a join index.

## 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.

## Performance hints 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 you 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 customer table (its primary key) and the cust\_id column of the sales\_order table. Therefore, you designate cust\_id in sales\_order as a FOREIGN KEY that references the id column of the customer table.

The first example creates the customer table, with the column id as its primary key. To simplify the example, other columns are represented here by ellipses (...).

```
CREATE TABLE DBA.customer
  ( id integer NOT NULL,
    ...
    PRIMARY KEY (id), )
```

Then you create the sales\_order table with six columns, specifying the column named id as the primary key. You also need to add a foreign key relating the cust\_id column of the sales\_order table to the id column of the customer 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 DBA.sales_order
  (id integer NOT NULL,
   cust_id integer NOT NULL
    REFERENCES DBA.customer(id),
   order_date date NOT NULL,
   fin-code-id char(2),
   region char(7),
   sales_rep integer NOT NULL,
   PRIMARY KEY (id), )
```

Alternatively, you could create the table without the REFERENCES clause, and then add the foreign key later, as is done in the following ALTER TABLE statement:

```
ALTER TABLE DBA.sales_order
  ADD FOREIGN KEY ky_so_customer (cust_id)
    REFERENCES DBA.customer (id)
```

A star join index has special requirements for table creation. See “Star joins” on page 302 for examples.

## 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.

## 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 Figure 6-1 on page 289, 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 Figure 6-1 you would need to use key joins. See “Types of join hierarchies” for more information.

---

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]: -----
```

## Issuing the CREATE JOIN INDEX statement

To create a join index, issue the CREATE JOIN INDEX statement. Here is a summary of the syntax for this command:

**CREATE JOIN INDEX** *join-index-name* **FOR** *join-clause*

The parameters of this command are:

*join-clause*:

[ ( ) *join-expression* *join-type* *join-expression*  
[ **ON** *search-condition* ] [ ) ]

*join-expression*:

{ *table-name* | *join-clause* }

*join-type*:

[ **NATURAL** ] **FULL** [ **OUTER** ] **JOIN**

*search-condition*:

[ ( ) *search-expression* [ **AND** *search-expression* ] [ ) ]

- The *join-clause* can be expressed either with or without parentheses.
- The ON clause can reference only two tables. One must be the current one, and the other can be any one table in the current join tree.

- All join predicates must be equijoins; that is, the *search\_expression* must indicate that the value in *column\_1* equals the value in *column\_2*. No single-variable predicates, intracolumn comparisons, or non-equality joins are permitted in the ON clause.
- To specify a multicolumn join, you include more than one predicate linking the two tables, and connect them with logical AND. Note that multicolumn indexes on base tables are *not* replicated in join indexes created using those base tables.
- You cannot connect join predicates with logical OR.
- The keyword NATURAL can replace the ON clause, when you are pairing columns from a single pair of tables by name.
- Join index tables must be IQ base tables. They must not be temporary tables, remote tables, or proxy tables.
- If a join column is a REAL data type, you must set the database option FLOAT\_AS\_DOUBLE to OFF when creating join indexes, or an error occurs. Issues may also result when using inexact numerics for join columns.

### Example 1: Key join

Here is an example of how you create a join index for the key join between the sales\_order table and the customer table. Remember that this is a key join, based on the foreign key ky\_so\_customer which relates the cust\_id column of sales\_order to the primary key id of the customer 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 customer FULL OUTER JOIN sales_order
```

### Example 2: ON clause join

The next example shows how you could create 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 customer FULL OUTER JOIN sales_order  
ON customer_id=sales_order.cust_id
```

### 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 customer to the cust\_id column of sales\_order, the following command would join the dissimilar id columns of the two tables:

```
CREATE JOIN INDEX customer_sales_order_join  
FOR customer NATURAL FULL OUTER JOIN sales_order
```

A natural join between the id columns of sales\_order and sales\_order\_items 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 sales_order NATURAL FULL OUTER JOIN  
sales_order_items
```

## Creating a join index in Sybase Central

To create a join index in Sybase Central, follow these steps.

❖ **To add a join index in Sybase Central:**

- 1 Select the Join Indexes folder in the left panel of the Sybase Central window.
- 2 Double-click the New Join Index icon (last on the toolbar).
- 3 <Unnamed> is highlighted in the Name box. Enter a name for the index.
- 4 From the Left Table Name dropdown, select a table name. Repeat for the Right Table Name.
- 5 Select a Join Type from the dropdown. If you select a type other than Natural, specify the Join Columns.
- 6 Click Advanced Properties to add a comment.
- 7 If you are only joining two tables, click Save and Close.
- 8 To join more than two tables, click Add Row. In the new row that appears, enter the next table to join in the Right Table Name column. Then click Save and Close.

## Types of join hierarchies

Sybase IQ supports two different types of join hierarchies:

- Linear joins
- Star joins

You create ad hoc joins for both linear and star joins. Join indexes are designed for use with linear joins.

## 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. In Figure 6-1 on page 289. 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

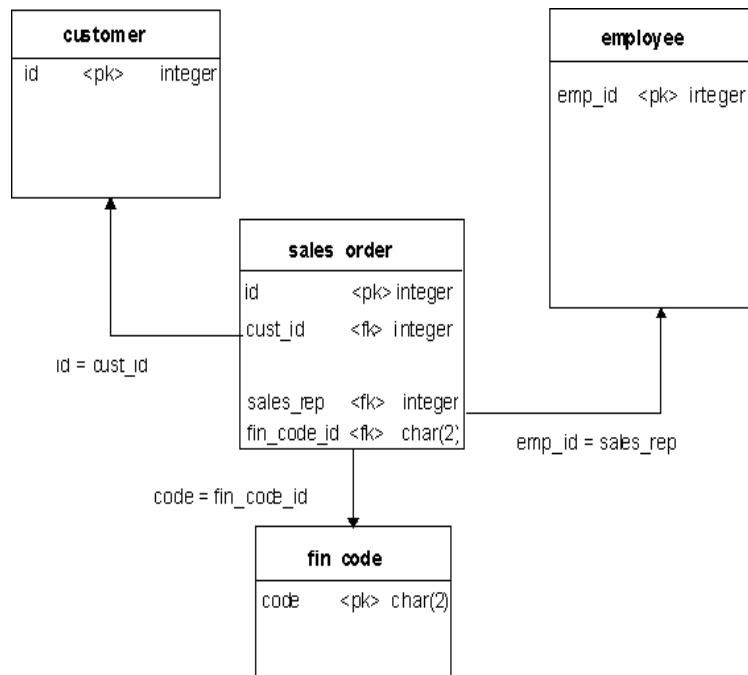
You can picture a star join as a structure with many branches, in which each branch is directly related to one table in the middle. In Figure 6-1, Tables D, F, and E form a very simple star join. More commonly, Table F would be at the center of many tables, each of which is joined to Table F.

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 sample database used throughout this book, the `sales_order` 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 "DBA"."sales_order" (
(
  "id"          integer NOT NULL,
  "cust_id"     integer NOT NULL
  REFERENCES "DBA"."customer" ("id"),
  "order_date"   datetime NOT NULL,
  "fin_code_id"  char(2) NULL
  REFERENCES "DBA"."fin_code" ("code"),
  "region"       char(7) NULL,
  "sales_rep"    integer NOT NULL
  REFERENCES "DBA"."employee" ("emp_id"),
  PRIMARY KEY ("id"),
);
  
```

As shown in the figure, the `sales_order` 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 sales_rep FROM sales_order  
WHERE sales_rep = 299
```

the results show 20 rows with 299 in the sales\_rep column.

However, if you enter:

```
SELECT emp_id FROM employee  
WHERE emp_id = 299
```

the results show only one row with 299 in the emp\_id 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.

To create a foreign key, see “Creating primary and foreign keys” on page 247. For other information on foreign keys, see “Declaring entity and referential integrity” on page 460.

## Modifying tables included in a join index

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.

## Inserting or deleting from tables in a join index

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.

You cannot perform partial-width inserts to tables that participate in a join index. If you need to add columns to a table in a join index, you must do one of the following:

- Drop the join index, do the partial-width insert, and then recreate the join index.

- Load or insert into all columns of the table.

## 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.

For more information on versioning, see Chapter 10, “Transactions and Versioning”

## Estimating the size and benefit of a join index

Before creating a join index, you should estimate its size and potential benefit.

### Using sp\_iqestjoin 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`.

See the *Sybase IQ Reference Manual* for syntax details of all stored procedures.

## Weighing join index benefit by comparing numbers of rows

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.

# Moving Data In and Out of Databases

## About this chapter

This chapter describes several methods of moving data into and out of your database and explains when you should use each of the methods. It also discusses conversion issues for data inserted from other types of databases.

## Contents

Topic	Page
Import and export overview	309
Exporting data from a database	313
Bulk loading data using the LOAD TABLE statement	324
Using the INSERT statement	351
Inserting specified values row by row	352
Inserting selected rows from the database	353
Importing data interactively	357
Inserting into tables of a join index	357
Inserting into primary and foreign key columns	358
Partial-width insertions	359
Converting data on insertion	364
Other factors affecting the display of data	381
Matching Adaptive Server Enterprise data types	382
Tuning bulk loading of data	387
Changing data using UPDATE	391
Deleting data	393

## 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 DBISQL 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 methods**

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 database, use the SQL statement `INSERT` with a `SELECT` statement clause.
- To remove specific rows from a table, use the `DELETE` statement.
- 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. See the section “Data extraction options” on page 315 for details and advantages of this feature.

From DBISQL 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, iq\_bcp, an open-client based utility that copies a database table to or from an operating system file in a user-specified format.

You can perform a BCP into an Anywhere table and then transfer the contents to Sybase IQ; however, the transfer of rows from Anywhere to Sybase IQ is executed one row at a time. Sybase IQ does not support BLKLIB, so BCP, which uses Open Client's Bulk-Library, doesn't work in load mode. Both Sybase IQ and Adaptive Server Enterprise BCP format supports a blank when one digit is in the date.

---

## Input and output data formats

The LOAD TABLE statement imports text files with one row per line. 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

File Format	Description
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 the section “Data extraction options” on page 315 for details on the output formats of data extraction.

## **Specifying an output format for Interactive SQL**

You can set the DBISQL output format in two ways:

- Select Command > Options from the DBISQL menu bar, and then choose an Output Format from the dropdown list. To make this the default output format, click Permanent.
- Specify the DBISQL option, OUTPUT\_FORMAT, to set the default output format.

For syntax details see the *Sybase IQ Reference Manual*.

## **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. For more information on permissions, see Chapter 12, “Managing User IDs and Permissions”.

## Scheduling 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 inserting into or deleting from different tables.

When you allow concurrent use of the database during updates, you pay a penalty in performance and disk use. For an explanation of how Sybase IQ handles concurrency issues, see Chapter 10, “Transactions and Versioning”. For other suggestions on improving load performance, see the section “Tuning bulk loading of data” on page 387.

## Exporting data from a database

This section tells how to export data from an Sybase IQ database.

---

**Note** To export IQ data from your database in this version of Sybase IQ, Sybase recommends that you use the methods described in this chapter. 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.

If you need to export tables (other than your system tables) from your Catalog Store, use the methods in this chapter, or see the *Adaptive Server Anywhere Reference Guide* for other ways to unload data.

---

## Using 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 DBISQL was started.

In this example, output is redirected to the file *empfile*:

```
SELECT *
FROM employee
># empfile
```

Do not enclose the file name in quotation marks.

Output redirection is most useful on the SELECT statement. Use the OUTPUT\_FORMAT option to control the format of the output file and the OUTPUT\_LENGTH option to control truncation. For example, the following commands set the format to ASCII text and does not truncate column contents:

```
SET OPTION OUTPUT_FORMAT = 'text'  
SET OPTION OUTPUT_LENGTH = 0
```

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 employee  
>& 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

The most common reason to extract data is for use in other software products. The other software package may not understand NULL values, however.

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 *Sybase IQ Reference Manual*.

## Data extraction options

The data extraction facility improves performance dramatically for queries with a large result set. This facility currently consists of a set of database options, which are set using the SET OPTION command. 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.

The extract options allow the user to redirect the output of a SELECT statement from the standard interface to go directly to one or more disk files or named pipes. 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

There are 27 options that control the behavior of extract (listed with allowed values for the option, followed by the default value):

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	"
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_Qualifier	ON or OFF	OFF
Temp_Extract_Qualifier_All	ON or OFF	OFF
Temp_Extract_Row_Delimiter	string	"
Temp_Extract_Size1	platform specific*	0
Temp_Extract_Size2	platform specific*	0

Option Name	Allowed Values	Default value
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\_Sizen 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 1TB 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 pathname. Extract starts extracting into a file with that name. Be sure to choose the pathname to a file that is not otherwise in use. If the file does not already exist, the data extraction facility creates the file.

Temp\_Extract\_Name1 is also used to specify the name of the output file, when the Temp\_Extract\_Append option is set ON. 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 pathname of an existing file and the Temp\_Extract\_Append option is set OFF (the default), the file contents will be 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\_Sizenn 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. 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. If Temp\_Extract\_Directory is set to FORBIDDEN for the PUBLIC group, then no one can run data extraction.

The Temp\_Extract\_Directory option provides increased security and helps control disk management by restricting the creation of large data extraction files to only 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\_Qoutes
- Temp\_Extract\_Qoutes\_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_Qoutes	string fields enclosed in quotes

Option	ASCII extraction action
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.

**Enabling the data extraction facility**

The data extraction options must be used 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 filename, 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.

---

**❖ Enabling data extraction options**

- 1 Save in a different file any old extract output you need to retain.
- 2 Remove any previously used extract output 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.  
`SET [ TEMPORARY ] OPTION option-name = option-value`
- 4 Issue a SELECT statement to extract the data you want.
- 5 Reset Temp\_Extract\_Name1 to the empty string, or disconnect if set temporarily, when you are done with extractions.

**Examples**

**Extracting to a single disk file** The following statements extract to the 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 file *daily\_report.txt*.

**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*.

**Extracting to multiple disk files** The following statements extract to disk files *file1.out*, *file2.out*, and *file3.out*.

First set the filename 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	<p>The following restrictions and limitations apply to the data extraction facility:</p> <ul style="list-style-type: none"><li>• Extract works only with data stored in the IQ Store.</li><li>• Extract does not work on system tables or cross database joins.</li><li>• Extract does not work with queries that use user-defined functions or system functions, except for the system functions <code>suser_id()</code> and <code>suser_name()</code>.</li><li>• A binary LOAD TABLE always trims blanks from VARCHAR data. If you have VARCHAR data with trailing blanks, they are not preserved on insert.</li><li>• Trailing zeros are padded onto VARBINARY data during the extract. For example, a field declared as <code>varbinary(6)</code>, which contains the data <code>0x1234</code>, is padded with zeros during extraction and is loaded after extraction as <code>0x123400</code>.</li><li>• Binary format will change in a future release.</li><li>• If you need to reproduce floating point data exactly, use the binary option.</li><li>• Tape devices are not supported at this time.</li><li>• If you run <code>dbisql</code> (Interactive SQL Java) with the <code>-q</code> (quiet mode) option and the data extraction commands are in a command file, you must first set and make permanent the <code>dbisql</code> option “Show multiple result sets.” If this option is not set, the output file is not created.</li></ul> <p>To set the “Show multiple result sets” option, click Tools &gt; Options in the <code>dbisql</code> window, then check the box “Show multiple result sets” and click “Make permanent.”</p> <p>Also note that when <code>Temp_Extract_Name1</code> is set, you cannot perform these operations:</p> <ul style="list-style-type: none"><li>• <code>LOAD</code>, <code>DELETE</code>, <code>INSERT</code> or <code>INSERT...LOCATION</code> to a table that is the top table in a join</li><li>• <code>SYNCHRONIZE JOIN INDEX</code> (issued explicitly or executed as part of <code>CREATE JOIN INDEX</code>)</li><li>• <code>INSERT...SELECT</code></li></ul>
Extraction and events	<p>Events do not support execution of statements that return result sets. The server log returns an error similar to the following:</p> <pre>Handler for event 'test_ev' caused SQLSTATE '09W03' Result set not permitted in 'test_ev'</pre>

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 = 'foo.out';
    SELECT * FROM iq_table;
END;

CREATE EVENT "test_ev" ENABLE HANDLER
BEGIN
    SELECT * INTO #tmp FROM proc1();
END;

TRIGGER EVENT test_ev;
```

Extraction with named pipes

Any UNIX process issuing a read request on a named pipe will wait forever until the process writing data to the pipe either sends data or an EOF (end-of-file). Each time the reading process receives data, it issues another read. If the writing process stops sending data and fails to send an EOF, the read process will wait in the kernel and cannot be interrupted from IQ.

Make sure that any process writing data into a named pipe always finishes with an EOF. If an IQ connection becomes unresponsive while writing out to a named pipe, try dumping the data out of the pipe. For example, issue the following commands from another thread:

```
cat NamedPipeFile > /dev/null
```

## Bulk loading data using the LOAD TABLE statement

The LOAD TABLE statement is used for efficient importing of data from a text or binary file into an existing database table. It loads data into any column indexes you have defined, as well as any created automatically.

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\_asiq. If the -gl option is set to ALL, 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.

See the description of the ON FILE ERROR load option for what happens when input file errors occur during a load.

#### Using command files to load data

To load large amounts of data, most users create command files. To create a command file, follow the instructions in the chapter entitled “Getting Started with DBISQL” in the *Introduction to Sybase IQ*.

#### 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 commit or neither.

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.

For more information on transaction processing, see Chapter 10, “Transactions and Versioning”.

#### Integrity constraints and LOAD TABLE

LOAD TABLE allows you to control load behavior when integrity constraints are violated and to selectively log information about the violations. 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.

For information on the contents and format of the message and row logs, see “Logging integrity constraint violations” on page 347.

#### Summary of LOAD TABLE syntax

The basic form of the LOAD TABLE statement is:

```
LOAD TABLE [ owner ].table-name
[ ( load-specification, ... ) ]
FROM 'filename-string', ...
[ FORMAT { 'ascii' | 'binary' } ]
... [ DELIMITED BY string ]
... [ STRIP { ON | OFF } ]
... [ QUOTES { ON | OFF } ]
```

```
... [ ESCAPES { ON | OFF } ]
... [ ESCAPE CHARACTER character ]
... [ WITH CHECKPOINT ON|OFF ]
... [ load-options ]
```

For usage and syntax details of all of the LOAD TABLE parameters, see “LOAD TABLE statement” in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

#### Load specification

The *load-specification* does the following:

- Lists each column to be loaded and describes the data in it. A column can contain fixed-length data, variable-length characters delimited by a separator, or data that uses a binary prefix to represent the number of bytes being read.
- Specifies FILLER format for any fields you want to skip.

The syntax for *load-specification* is as follows:

```
load-specification:
{ column-name [ column-spec ] |
  FILLER ( filler-type ) }
```

For each column, you can specify a *column-spec*. If you omit this option, the format information in the *load-options* applies to this column. The *column-spec* and *load-options* format information tell Sybase IQ what type of data to expect, and how to convert it into a compatible data format if necessary.

Syntax for the *column-spec* is:

```
column-spec:
{ ASCII ( input-width ) |
  BINARY [ WITH NULL BYTE ] |
  PREFIX { 1 | 2 | 4 } |
  'delimiter-string' |
  DATE ( input-date-format ) |
  DATETIME ( input-datetime-format ) |
  [ NULL ( { BLANKS | ZEROS | 'literal', ... } ) ] }
```

You can specify the following types of data in the *column-spec*:

- Data with bytes of fixed length. Although specified by the keyword ASCII, any 8-bit characters may be used, and for 16-bit character sets, two 8-bit characters are used for each 16-bit character. No code conversion is performed for char and varchar fields except truncation, blank stripping, or blank padding. ASCII is also used to fill numeric data, time, and datetime fields. In each case, the conversion is the same if the value were first inserted as a character field, then cast to the data type of the column in the table. The *input-width* value is an integer value indicating the fixed width in bytes of the input field in every record.

- Binary fields that use a number of PREFIX bytes (1, 2, or 4) to specify the length of the binary input. The BINARY keyword indicates that data is already converted to the internal form (except for when the byte-order load option is specified).

Note that if the data is unloaded using the extraction facility with the TEMP\_EXTRACT\_BINARY option set ON, then you *must* use the BINARY WITH NULL BYTE parameter for each column when you load the binary data.

- Variable-length characters delimited by a separator. You specify the *delimiter-string* as a string of one to four ASCII characters, or any 8-bit hexadecimal ASCII code that represents a single, non-printing character. The delimiter-string must be enclosed in single quotes. For example, you specify:
  - '\x09' to represent a tab as the terminator.
  - '\x00' for a null terminator (no visible terminator as in "C" strings).
  - '\x0a' for a newline character as the terminator. You can also use the special character combination of \n for newline.
- DATE or DATETIME string as ASCII characters. You must define the *input-date-format* or *input-datetime-format* of the string using one of the corresponding formats for the date and datetime data types supported by Sybase IQ. Use DATE for date values and DATETIME for datetime and time values. For information about these data types, see the *Sybase IQ Reference Manual*.

---

**Note** The *column-spec* is for IQ tables only. If you specify a *column-spec* for a Catalog Store table, you get an error.

---

The NULL portion of the *column-spec* indicates how to treat certain input values as NULL values when loading into the table column. These characters can include BLANKS, ZEROS, or any other list of literals you define. When you specify a NULL value or read a NULL value from the source file, the destination column must be able to contain NULLs.

ZEROS are interpreted as follows: the cell is set to NULL if (and only if) the input data (before conversion, if ASCII) is all binary zeros (and not character zeros).

- If the input data is character zero, then:
  - a) NULL(ZEROS) never causes the cell to be NULL

- b) NULL('0') causes the cell to be NULL
- If the input data is binary zero (all bits clear), then:
  - a) NULL(ZEROS) causes the cell to be NULL
  - b) NULL('0') never causes the cell to be NULL

For example, if your LOAD statement includes `col1 date('yymmdd')` `null(zeros)` and the date is 000000, you will receive an error indicating that 000000 cannot be converted to a DATE(4). To get the load statement to insert a NULL value in col1 when the data is 000000, you must write the NULL clause as `null('000000')`, or modify the data to equal binary zeros and use NULL(ZEROS).

The FILLER clause indicates you want to skip over a specified field in the source input file. For example, there may be characters at the end of rows or even entire fields in the input files that you do not want to add to the table. As with the *column-spec* definition, FILLER allows you to specify ASCII fixed length of bytes, variable length characters delimited by a separator, and binary fields using PREFIX bytes. The maximum length of a variable-length FILLER column is 512 bytes. FILLER clause syntax is as follows:

```
FILLER ( filler-type )
filler-type:
{ input-width | PREFIX { 1 | 2 | 4 } | 'delimiter-string' }
```

For more information on how to use data conversion options, see “Converting data on insertion” on page 364.

#### Specifying files to load

You specify one or more files from which to load data. In the FROM clause, you specify each *filename-string*, 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 is rolled back.

**filename-string**    The *filename-string* is passed to the server as a string. The string is therefore subject to the same formatting requirements as other SQL strings. 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.) Therefore, the statement to load data from the file *c:\newinput.dat* into the employee table is:

```
LOAD TABLE employee  
FROM 'c:\\newinput.dat' ...
```

- The pathname 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.

#### Named pipes

The file specification can be a named pipe. 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 you do 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.

#### Specifying table-wide format options

You can specify several options that describe the format of input data.

**FORMAT option** You can specify a default format for table columns, which applies if you omit the *column-spec*. The same formats that can appear in the *column-spec* can appear here. If you also omit the **FORMAT** load option, the file is assumed to be binary.

**DELIMITED BY option** If you omit a column delimiter in the *column-spec* definition, the default column delimiter character is a comma. You can specify an alternative column delimiter by providing a single ASCII character or the hexadecimal representation. In particular, to specify tab-delimited values use the hexadecimal ASCII code of the tab character (9), as follows:

```
...DELIMITED BY '\\x09' ...
```

To use the newline character as a delimiter, you can specify either the special combination '\n' or its ASCII value '\x0a'.

---

**Note** Although the *delimiter-string* in the *column-spec* may be a string of up to four characters, the **DELIMITED BY** option allows only a single ASCII character or its hexadecimal representation.

---

**STRIP option** With STRIP turned on (the default), trailing blanks are stripped from values before they are inserted. This is effective only for VARCHAR data. To turn the STRIP option off, enter the clause as follows:

```
...STRIP OFF ...
```

Trailing blanks are stripped only for non-quoted strings. Quoted strings retain their trailing blanks. If you don't require blank sensitivity, you may use the FILLER option allows you to be more specific in the number of bytes to strip instead of just all the trailing spaces.

This option does not apply to ASCII fixed-width inserts. For example, the STRIP option in the following statement is ignored:

```
LOAD TABLE dba.foo (col1 ascii(3), col2 ascii(3))
  FROM foo_data QUOTES OFF ESCAPES OFF STRIP ON
```

**QUOTES option** The QUOTES parameter is optional and the default is ON. With QUOTES turned on, LOAD TABLE expects input strings to be enclosed in quote characters. The quote character is either an apostrophe (single quote) or a quotation mark (double quote). The first such character encountered in a string is treated as the quote character for the string. String data must be terminated with a matching quote.

With QUOTES ON, column or row delimiter characters can be included in the column value. Leading and ending quote characters are assumed not to be part of the value and are excluded from the loaded data value.

With QUOTES OFF, Sybase IQ does not strip off apostrophes (single quotes) or quotation marks (double quotes). When it encounters these characters in your input file, it treats them as part of the data. With QUOTES OFF, you cannot include column delimiter characters in column values.

For syntax and usage details, see “LOAD TABLE statement” in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

**QUOTES option example** 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
```

The following output displays the result of the query enclosed by '<' and '>' and the values of the LOAD TABLE options QUOTES and STRIP:

<b>QUOTES option</b>	<b>STRIP option</b>	<b>c1</b>	<b>c2</b>	<b>c3</b>	<b>length(c2)</b>	<b>length(c3)</b>
ON	ON	<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	ON	<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 ON, 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 ON, 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 ON, both leading space and trailing space within quotes for the enclosed fields c2 and c3 row 3 are NOT trimmed.

**ESCAPES option** Currently, you must specify ESCAPES OFF. The default of ESCAPES ON is provided for compatibility with Adaptive Server Anywhere; this option may be supported in a future version. With ESCAPES turned on, if you omit a *column-spec* definition for an input field, characters following the backslash character are recognized and interpreted as special characters by the database server. Newline characters can be included as the combination \n, and other characters can be included in data as hexadecimal ASCII codes, such as \x09 for the tab character. A sequence of two backslash characters (\\\ ) is interpreted as a single backslash.

**WORD SKIP option** Allows the load to continue when it encounters data longer than the limit specified when the word index was created. For details, see “LOAD TABLE statement,” Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

#### Example

The following UNIX example specifies a BLOCK FACTOR of 50,000 records along with the PREVIEW option:

```
LOAD TABLE lineitem
      (l_shipmode ASCII(15),
       l_quantity ASCII(8),
       FILLER(30))
  FROM '/d1/MILL1/tt.t'
  BLOCK FACTOR 50000 PREVIEW ON
```

#### Specifying load options

You can specify a wide range of load options. These options 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. Syntax for *load-options* is as follows:

```
...[ { BLOCK FACTOR number | BLOCK SIZE number } ]
...[ BYTE ORDER { NATIVE | HIGH | LOW } ]
...[ LIMIT number-of-rows ]
...[ NOTIFY number-of-rows ]
...[ ON FILE ERROR { ROLLBACK | FINISH | CONTINUE } ]
...[ PREVIEW { ON | OFF } ]
...[ ROW DELIMITED BY 'delimiter-string' ]
...[ SKIP number-of-rows ]
...[ START ROW ID number ]
...[ UNLOAD FORMAT ]
...[ ON PARTIAL INPUT ROW { ROLLBACK | CONTINUE } ]
...[ IGNORE CONSTRAINT constrainttype [, ...] ]
...[ MESSAGE LOG 'string' ROW LOG 'string'
    [ ONLY LOG logwhat [, ...] ] ]
...[ LOG DELIMITED BY 'string' ]
```

The syntax for the parameters is as follows:

```
constrainttype:  
{ CHECK integer | UNIQUE integer  
| NULL integer  
| FOREIGN KEY integer  
| DATA VALUE integer  
| ALL integer }  
  
logwhat:  
{ CHECK | ALL | NULL | UNIQUE | DATA VALUE | FOREIGN KEY }
```

Each of these options is described briefly below. For details of all options of the LOAD TABLE statement, see the *Sybase IQ Reference Manual*.

**BLOCK FACTOR option** Specifies blocking factor, or number of records per block, used when a source was created. This option is not valid for insertions from variable length input fields; use the BLOCK SIZE option instead. However, it does affect all file inserts (including from disk) with fixed length input fields, and it can affect performance dramatically.

The default setting for BLOCK FACTOR is 10,000. Higher block factors generally improve the speed of I/O operations. However, consider the following when setting this option:

- If your source is a disk file, memory considerations will determine the best setting for your system.
- If your source is a tape, either use the same blocking factor that was used when creating the tape (for best performance) or a blocking factor that is evenly divisible into it.
- Sybase IQ rejects the insert operation if you specify a BLOCK FACTOR of zero.
- You cannot specify BLOCK FACTOR along with BLOCK SIZE or with any variable-width input fields.
- If you are loading very wide varchar data, reduce the BLOCK FACTOR to 10.

**ESCAPE CHARACTER option** Specifies an alternative escape character. The default escape character for characters stored as hexadecimal codes and symbols is a backslash (\), so that \x0A is the linefeed character, for example.

This can be changed using the ESCAPE CHARACTER clause. For example, to use the exclamation mark as the escape character, you would enter:

```
... ESCAPE CHARACTER '!'
```

Only one single-byte character can be used as an escape character.

---

**Note** Because you must specify ESCAPES OFF in this version of Sybase IQ, the ESCAPE CHARACTER option has no effect. It is provided for compatibility with Adaptive Server Anywhere.

---

**WITH CHECKPOINT ON clause** If this option is set to ON, a checkpoint is issued when the LOAD TABLE statement completes and is logged. In the event recovery is required, it is guaranteed even if the data file is then removed from the system.

If WITH CHECKPOINT ON is not specified, the file used for loading must be retained in case recovery is required.

**BLOCK SIZE option** Specifies the default size in bytes in which input should be read. This option only affects variable-length input data read from files; it is not valid for fixed-length input fields. It is similar to BLOCK FACTOR, but there are no restrictions on the relationship of record size to block size. You cannot specify this option along with the BLOCK FACTOR option.

The default setting for BLOCK SIZE is 500,000, which is high enough for input from disk files. For tape files, you should specify the same block size that was used when creating the tape. You cannot specify BLOCK SIZE along with BLOCK FACTOR or with any fixed width input fields.

#### Example

The following UNIX example specifies a BLOCK SIZE of 200,000 bytes:

```
LOAD TABLE mm
    (l_orderkey '\x09',
     l_quantity '\x09',
     l_shipdate DATE('YYYY/MM/DD'))
  FROM '/d1/MILL1/tt.t'
  BLOCK SIZE 200000
```

**BYTE ORDER option** Specifies the byte ordering during reads. This option applies to all binary input fields, including those defined as PREFIX 2 or PREFIX 4. If none are defined, this option is ignored. Sybase IQ always reads prefix binary data in the format native to the machine it is running on (default is NATIVE). You can also specify:

- HIGH when multibyte quantities have the high order byte first (for big endian platforms like Sun, IBM AIX, and HP).
- LOW when multibyte quantities have the low order byte first (for little endian platforms like Windows).

#### Example

Here 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
```

**LIMIT option** Specifies the maximum number of rows to insert into the table. The default is 0 for no limit. The maximum is 2GB-1.

LIMIT works together with the SKIP option. SKIP indicates where to begin reading from the input file, and LIMIT specifies how many of those rows to insert. SKIP takes precedence over LIMIT. If you specify multiple input files, *these options only affect the first file*. The following table shows how these options work together:

**Table 7-1: SKIP and LIMIT insert options**

If the SKIP value is	And the LIMIT value is	Then IQ does this
0	5	Reads 5 rows and inserts 5 rows.
20	5	Reads 25 rows and inserts 5 rows.
10	5	Reads 10 rows and inserts 5 rows. If the input file has only 8 rows, then zero rows are 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'
BLOCK FACTOR 1000
PREVIEW ON
LIMIT 1000000
```

**NOTIFY option** Specifies that you be notified with a message each time the specified number of rows is inserted successfully into the table. The default is every 100,000 rows. Very frequent notifications can slow down your insert operation. To turn off NOTIFY entirely, set NOTIFY = 0. See “Interpreting notification messages” for an explanation of messages.

**ON FILE ERROR option** Specifies the action Sybase IQ takes when an input file cannot be opened, either because it does not exist or because you have incorrect permissions to read the file. For all other reasons or errors, it aborts the entire insertion. You can specify one of the following:

- ROLLBACK aborts the entire transaction (the default).
- FINISH finishes the insertions already completed and ends the load operation.
- CONTINUE returns an error but only skips the file to continue the load operation. You cannot use this option with partial-width inserts.

**PREVIEW option** Displays the layout of input into the destination table including starting position, name, and data type of each column. Sybase IQ displays this information at the start of the load process. If you are writing to a log file, this information is also included in the log.

This option is especially useful with partial-width inserts. It can help you diagnose failed or skewed insertions due to incompatible data types, or destination column alignment that does not match source columns. Look at the expected column data type and starting position information to determine if you need to use an insert conversion option on a column and/or where and how much filler to use.

---

**Note** PREVIEW ON helps you determine if a load is correct. It does not stop the load from occurring.

---

**ROW DELIMITED BY option** Specifies a string up to 4 bytes in length that indicates the end of an input record. You can use this option only if all fields within the row are any of the following:

- Delimited with column terminators
- Data defined by the DATE or DATETIME *column-spec* options
- ASCII fixed length fields

The row delimiter can be any string of from 1 to 4 8-bit codes, including any combination of printable characters, and/or any 8-bit hexadecimal code that represents a non-printing character. For example, you specify \x09 to represent a tab as the terminator. For a null terminator (no visible terminator as in "C" strings), you specify \x00.

To use the newline character as a row delimiter, you can specify either the special combination '\n' or its ASCII value '\x0a'.

You cannot use this option if any input fields contain binary data. With this option, a row terminator causes any missing fields to be set to NULL. All rows must have the same row delimiters, and it must be distinct from all column delimiters. The row and field delimiter strings cannot be an initial subset of each other. For example, you cannot specify “\*” as a field delimiter and “\*#” as the row delimiter, but you could specify “#” as the field delimiter with that row delimiter.

If a row is missing its delimiters, Sybase IQ returns an error and rolls back the entire load transaction. The only exception is the final record of a file where it rolls back that row and returns a warning message.

On Windows, a row delimiter is usually indicated by the newline character followed by the carriage return character. You may need to specify this as the *delimiter-string* for either this option or FILLER.

#### Example

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'
```

**SKIP option** Lets you define a number of rows to skip at the beginning of the input file(s) for this load. The default is 0. This option works in conjunction with the LIMIT option, and takes precedence over it.

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'
BLOCK FACTOR 1000
LIMIT 4000
SKIP 5000
PREVIEW ON
```

**START ROW ID option** Specifies the id number of a row in the table where insertions should begin. This option is used for *partial-width insertions*, which insert into a subset of the columns in the table. If you are inserting data into an existing row, you must define the format of each input column with a *column-spec*, and use START ROW ID to identify the row where you want to insert it. The default is 0, which causes data to be inserted in a new row wherever there is space in the table. Be sure to read “Partial-width insertions” before using this option and performing partial-width inserts.

**UNLOAD FORMAT option** Specifies that the data in the input file is in the format produced by the UNLOAD command in Sybase IQ 11.5.1, specifically for upgrading to Sybase IQ 12.x. This format places certain restrictions on other load options you specify:

- The format in the column specifications must be BINARY, the default. Specifying ASCII, PREFIX, FILLER, or *string-delimiter* causes an error.
- You must not use the load options DELIMITED BY and ROW DELIMITED BY.
- To allow NULLs in the data you must specify BINARY WITH NULL BYTE in the column specification. You cannot include NULL in the *column-spec* in any other way.
- For the sake of consistency with the data being loaded, you can specify BINARY WITH NULL BYTE even when loading into a table column that does not allow NULLs (as specified in CREATE TABLE or ALTER TABLE). However, if you try to load any data into a column that does not allow NULLs, you receive an error.

See the *Sybase IQ Installation and Configuration Guide* for more information on upgrading.

**ON PARTIAL INPUT ROW option** Specifies the action to take when a partial input row is encountered during a load. You can specify one of the following:

- CONTINUE issues a warning and continues the load operation. This is the default.
- ROLLBACK aborts the entire load operation and reports the error

```
Partial input record skipped at EOF.  
SQLSTATE: QDC32      SQLSTATE: -1000232L
```

**IGNORE CONSTRAINT** option   Specifies whether to ignore UNIQUE, NULL, CHECK, DATA VALUE, and/or FOREIGN KEY integrity constraint violations that occur during a load and the maximum number of violations to ignore before initiating a rollback. Specifying each *constrainttype* has the following result:

- **UNIQUE *limit*** If *limit* specifies zero, then the number of UNIQUE constraint violations to ignore is infinite. If *limit* is non-zero, then the *limit* +1 occurrence of a UNIQUE constraint violation causes the load to rollback.
- **NULL *limit*** If *limit* specifies zero, then the number of NULL constraint violations to ignore is infinite. If *limit* is non-zero, then the *limit* +1 occurrence of a NULL constraint violation causes the load to rollback.
- **FOREIGN KEY *limit*** If *limit* specifies zero, then the number of FOREIGN KEY constraint violations to ignore is infinite. If *limit* is non-zero, then the *limit* +1 occurrence of a FOREIGN KEY constraint violation causes the load to rollback.
- **DATA VALUE *limit*** If the database option CONVERSION\_ERROR = ON, then an error is reported and the statement rolls back. If *limit* specifies zero, then the number of DATA VALUE constraint violations (data type conversion errors) to ignore is infinite. If *limit* is non-zero, then the *limit* +1 occurrence of a DATA VALUE constraint violation causes the load to rollback.
- **ALL *limit*** If the database option CONVERSION\_ERROR = ON, then an error is reported and the statement rolls back. If *limit* specifies zero, then the cumulative total of all integrity constraint violations to ignore is infinite. If *limit* is non-zero, then load rolls back when the cumulative total of all ignored UNIQUE, NULL, DATA VALUE, and FOREIGN KEY integrity constraint violations exceeds the value of *limit*. For example, you specify the following IGNORE CONSTRAINT option:

```
IGNORE CONSTRAINT NULL 50, UNIQUE 100, ALL 200
```

The total number of integrity constraint violations cannot exceed 200, while the total number of NULL and UNIQUE constraint violations cannot exceed 50 and 100, respectively. Whenever any of these limits is exceeded, the LOAD TABLE statement rolls back.

---

**Note** A single row can have more than one integrity constraint violation. Every occurrence of an integrity constraint violation counts towards the limit of that type of violation.

Sybase strongly recommends setting the 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.

- 
- **CHECK***limit* If *limit* specifies zero, then the number of check constraint violations to ignore is infinite. If *limit* is non-zero, then the *limit + 1* occurrence of a CHECK constraint violation causes the load to roll back.

---

**Note** A single row can have more than one integrity constraint violation. Every occurrence of an integrity constraint violation counts towards the limit of that type of violation.

Sybase strongly recommends setting the 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.

---

If UNIQUE, NULL, CHECK, or FOREIGN KEY is not specified in the IGNORE CONSTRAINT clause, then the load rolls back on the first occurrence of each of these types of integrity constraint violation.

If DATA VALUE is not specified in the IGNORE CONSTRAINT clause, then the load rolls back on the first occurrence of this type of integrity constraint violation, unless the database option CONVERSION\_ERROR = OFF. If CONVERSION\_ERROR = OFF, then a warning is reported for any DATA VALUE constraint violation and the load continues.

When the load completes, an informational message regarding integrity constraint violations is logged in the .iqmsg file. This message contains the number of integrity constraint violations that occurred during the load and the number of rows that were skipped.

**MESSAGE LOG option** Specifies the names of files in which to log information about integrity constraint violations and the types of violations to log. Timestamps indicating the start and completion of the load are logged in both the MESSAGE LOG and the ROW LOG files. Both MESSAGE LOG and ROW LOG must be specified, or no information about integrity violations is logged.

- If the ONLY LOG clause is not specified, then no information on integrity constraint violations is logged. Only the timestamps indicating the start and completion of the load are logged.
- Information is logged on all integrity constraint type violations specified in the ONLY LOG clause.
- If constraint violations are being logged, then every occurrence of an integrity constraint violation generates exactly one row of information in the MESSAGE LOG file.

The number of rows (errors reported) in the MESSAGE LOG file can exceed the IGNORE CONSTRAINT option limit, because the load is performed by multiple threads running in parallel. More than one thread may report that the number of constraint violations has exceeded the specified limit.

- If constraint violations are being logged, then exactly one row of information is logged in the ROW LOG file for a given row, regardless of the number of integrity constraint violations that occur on that row.

Note that 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 described above for the MESSAGE LOG.

- The MESSAGE LOG and ROW LOG files cannot be raw partitions.
- If the MESSAGE LOG or ROW LOG file already exists, then new information is appended to the file.
- Specifying an invalid filename for the MESSAGE LOG or ROW LOG file generates an error.
- Specifying the same filename for the MESSAGE LOG and ROW LOG files generates an error.

Various combinations of the IGNORE CONSTRAINT and MESSAGE LOG options result in different logging actions, as indicated in the following table:

IGNORE CONSTRAINT specified?	MESSAGE LOG specified?	Action
yes	yes	All ignored integrity constraint violations are logged, including the user specified limit, before the rollback.
no	yes	The first integrity constraint violation is logged before the rollback.
yes	no	Nothing is logged.
no	no	Nothing is logged. The first integrity constraint violation causes a rollback.

**Note** Sybase strongly recommends setting the IGNORE CONSTRAINT option limit to a non-zero value, if you are logging the ignored integrity constraint violations. If a single row has more than one integrity constraint violation, a row for *each* violation is written to the MESSAGE LOG file. Logging an excessive number of violations affects the performance of the load.

**LOG DELIMITED BY option** Specifies the separator between data values in the ROW LOG file. The default separator is a comma.

For more details on the contents and format of the MESSAGE LOG and ROW LOG files, see “Logging integrity constraint violations” on page 347.

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. See “Partial-width insertions” for examples of how you use this option to insert data into existing rows.

If you want to empty out an existing table and reload it, you can use the TRUNCATE TABLE statement to remove all the rows from a table.

Simple LOAD TABLE 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 department table.

```
LOAD TABLE department
  FROM 'dept.txt'
```

## 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 this one:

```
In table 'partsupp', the full width insert  
of 5 columns will begin at record 1.  
2001-02-27 13:03:47 0000000002 Insert Started:  
2001-02-27 13:03:47 0000000002 partsupp  
2001-02-27 13:03:48 0000000002 [20270] :  
=n*** File: /remote/rip/tpcd_data/scale_1/partsupp.tbl
```

Each time it inserts the number of records specified in the NOTIFY load option, Sybase IQ sends a message like this:

```
2001-02-27 13:03:49 0000000002 [20897]: 100000 Records, 2 Seconds  
Mem: 469mb/M470  
Main Blks: U63137/6%, Buffers: U12578/L7  
Temporary Blks: U273/0%, Buffers: U1987/L1960  
Main I: L331224/P22 O: D25967/P7805 C:D0  
Temporary I: L25240/P8 O: D4749/P0 C:D0
```

The first line shows how many rows Sybase IQ has read since the last notification message, and the number of seconds taken reading them. Even if Sybase IQ reads the same number of messages each time, the amount of time varies depending on the data read (for example, how many data conversions are required). Reported time intervals smaller than 1 second are usually reported as “0 Secs”.

## Memory message

This message displays memory usage information:

```
Mem: 469mb/M470
```

**Table 7-2: Memory messages**

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 messages

This line describes the permanent (main) IQ Store:

Main            Blks: U63137/6%, Buffers: U12578/L7

**Table 7-3: Main blocks**

Item	Description
U#	Number of blocks in use.
#%	Percentage of database filled.
Buffers: U#	<p>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.</p> <p><i>Note:</i> 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.</p>
L#	<p>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.</p> <p>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.</p> <p><b>Note</b> 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.</p>

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” in Chapter 1, “Troubleshooting Hints,” of the *Sybase IQ Troubleshooting and Recovery Guide*.

## IQ Temporary Store blocks message

This message provides similar information to the Main IQ Store Blocks message explained above.

Temporary Blks: U273/0%, Buffers: U1987/L1960

## Main buffer cache activity message

This line displays information about the main buffer cache.

```
Main      I: L331224/P22 O: D25967/P7805 C:D0
```

**Table 7-4: Main IQ Store file message**

Item	Description
Main: I: L#	Number of logical file reads.
P#	Number of physical file reads.
O: D#	Number of times a buffer was destroyed.
P#	Number of physical writes.
C: D#	Buffer manager data compression ratio. This is the total number of bytes eligible for compression minus 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. If either of these conditions exists, see *Sybase IQ Performance and Tuning Guide* for information on setting buffer cache size or using the IQ performance monitor.

## Temporary buffer cache message

These lines display information about the Temp buffer cache.

```
Temporary I: L25240/P8 O: D4749/P0 C:D0
```

See the description for the Main buffer cache message above.

## 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. They appear in the *.iqmsg* file only once per database connection. 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” in the *Sybase IQ Troubleshooting and Recovery Guide*.

---

```
2002-11-12 09:34:42 0000000002 Txn 173
2002-11-12 09:34:42 0000000002 Connect: 1550990889. SA connID: 1. User: DBA.
```

## Controlling message logging

The *NOTIFY\_MODULUS* database option adjusts the default frequency of notification messages during loads, or omits these messages. The *NOTIFY* option in the *LOAD* command overrides the *NOTIFY\_MODULUS* setting.

The *IQMSG\_LENGTH\_MB* database option enables message log wrapping.

See Chapter 2, “Database Options” in the *Sybase IQ Reference Manual* for details of these database options.

## Logging integrity constraint violations

*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*).

## 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 and trailer appear once per load. 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:

```
2002-02-24 23:04:31 LOAD TABLE customer Integrity Constraint Violations
```

The row information message consists of three parts:

- **rowid** The row number within the table where this row would have been loaded, if an integrity constraint violation had not occurred.
- **type** The type of integrity constraint violation detected.
- **column numbers** 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:

```
2002-02-24 23:05:43 LOAD TABLE customer 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 and trailer appear once per load.

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:

```
2002-02-24 23:04:31 LOAD TABLE customer 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:

- **rowid** The row number within the table where this row would have been loaded, if an integrity constraint violation had not occurred.
- **data values** 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:

```
2002-02-24 23:05:43 LOAD TABLE customer 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 customer(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 customer table:

```
LOAD TABLE customer ...  
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:

```
2002-02-24 23:04:31 LOAD TABLE customer Integrity Constraint Violations  
1267 DATA VALUE 4  
3216 UNIQUE 1  
3216 NULL 6  
9677 NULL 1  
2002-02-24 23:05:43 LOAD TABLE customer Completed
```

After the LOAD TABLE completes, the ROW LOG file *row.log* contains the following information:

```
2002-02-24 23:04:31 LOAD TABLE customer 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#
```

```
2002-02-24 23:05:43 LOAD TABLE customer Completed
```

## 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.

## Inserting specified values row by row

To add specified values to a table row by row, use the `INSERT` statement with this syntax:

```
INSERT [ INTO ]
[ owner.] table_name
[ (column_name, ...) ]
... VALUES (expression...)
```

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. See "Converting data on insertion" for information on data type conversions. See the *Sybase IQ Reference Manual* for a complete description of Sybase IQ data types.

---

**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.

## Inserting selected rows from the database

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 this syntax:

```
INSERT [ INTO ]
[ owner.]table_name
[ (column-name,...) ]
[ insert-load-options ]...
select-statement

insert-load-options:
LIMIT number-of-rows
NOTIFY number-of-rows
SKIP number-of-rows
START ROW ID number
```

This form of the INSERT statement lets you insert any number of rows of data, based on the results of a general SELECT 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. See "Converting data on insertion" for more information about data types and conversion options.

With this form of the INSERT statement you can specify any of the following insert-load-options:

The START ROW ID option lets you perform a partial-width insert. Read "Partial-width insertions" before you specify this option.

For an explanation of all of these options, see "Specifying 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. For details, see Chapter 16, "Accessing Remote Data" and Chapter 17, "Server Classes for Remote Data Access."

#### Inserting directly from an Adaptive Server Enterprise database

You can insert data easily from an Adaptive Server Enterprise or SQL Server database, using the LOCATION syntax of the INSERT statement. You can also use this method to move selected columns from a pre-Version 12 Sybase IQ database into a Version 12.x database.

In order to use this capability, 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 pre-Version 12 IQ database, and INSERT permission on the target IQ 12.x database

❖ **Inserting data directly from Adaptive Server Enterprise**

- 1 Connect to both the Adaptive Server Enterprise and the Sybase IQ 12.x database using the same user ID and password.
- 2 On the Sybase IQ 12.x database, issue a statement using this syntax:

```
INSERT INTO asiq_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. For more information and an example, see “*INSERT statement*” in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

Loading ASE text and images

Sybase IQ does not support the Adaptive Server Enterprise data type TEXT, but you can execute INSERT...LOCATION from an ASE database column of data type TEXT. Also note that INSERT...LOCATION does not support the use of variables in the SELECT statement. Text and image data inserted is right truncated at 32767 bytes.

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.)

If you need to load larger data, see “*Bulk loading data using the LOAD TABLE statement*” on page 324.

For details on the syntax of the *INSERT* statement, see Chapter 6, “SQL Statements,” of the *Sybase IQ Reference Manual*.

Example

The following command inserts data from the l\_shipdate and l\_orderkey columns of the lineitem table from the Sybase IQ database asiq11db.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.asiq11db'  
{ 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.asiqdb'  
PACKETSIZE 512  
{ SELECT l_shipdate, l_orderkey  
FROM lineitem  
WHERE l_orderkey = 1 }
```

---

**Note** If you use `START ROW ID` and you select fewer columns than exist in the destination table, the columns in remaining rows of the destination table will be NULLs, if NULLs are legal values. See “Partial-width insertions” on page 359 for more information.

---

#### Importing data from pre-Version 12 Sybase IQ

To import data from an Sybase IQ database version earlier than 12.0, you must use one of the following methods:

- The `LOAD TABLE` command with the `UNLOAD FORMAT` option
- The `INSERT...LOCATION` syntax

You cannot use other forms of the `INSERT` command.

For more information on loading from an older version, see the *Sybase IQ Installation and Configuration Guide*.

## Importing data interactively

If you are inserting small quantities of data, you may prefer to enter it interactively through DBISQL, 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 ( . . . )
```

For more information about the `INSERT` command, see “Using the `INSERT` statement”.

---

**Note** Do not use the Import option in the DBISQL Data menu. This option is not supported for use with Sybase IQ databases.

---

## 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 are only two differences:

- The data in a join index must be synchronized before you can use the join index to resolve queries.
- You cannot perform a partial-width insert for tables that participate in a join index.

---

**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` command. For the syntax of this command, see “Synchronizing join indexes” or see the *Sybase IQ Reference Manual*.

Updating from  
different connections  
may cause errors

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.

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.

For more information on join indexes, see Chapter 6, “Using Sybase IQ Indexes” For more information on transaction processing, see Chapter 10, “Transactions and Versioning”

## **Inserting 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.

## Partial-width insertions

By default, new rows are inserted wherever there is space in the indexes, and each LOAD TABLE or INSERT statement starts a new row. This approach works as long as the data you are inserting is a new row. Sybase IQ also lets you insert individual columns into an existing row, if you specify its rowid.

A *partial-width insertion*, also called a vertical insertion, is an insertion into a subset of columns in a table. You can use two or more partial-width insertions to insert data into all of the columns of the table.

Partial-width insertions let you:

- Insert data into just a few columns at a time. This approach can be helpful if you have memory limitations.

For example, you can insert data into a few columns at a time, using separate LOAD TABLE or INSERT statements for each group of indexes and using the START ROW ID option to keep the ROW IDs consistent and the memory requirement lower. You may want to do this if you are inserting into a very wide table and do not have enough free memory to populate all the indexes at one time.

- Use different data sources, such as multiple flat files, to insert into different groups of columns in a table.
- Add a new column and corresponding index to a table after you have already inserted data into the columns for that table. For more information, see the ALTER INDEX command.

---

**Warning!** This is an advanced operation. If you do not perform all the steps correctly in a partial-width insert, you may insert data incorrectly. Never use this type of insert unless you are an experienced Sybase IQ user and are very familiar with your data. Full-width inserts, which insert into all the column indexes on a table at the same time, ensure row-level integrity and are less error-prone.

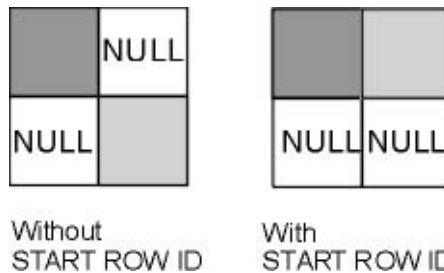
---

Use START ROW ID to specify at which row you want to start the insert. This allows you to insert into some of the columns in a row with one partial-width INSERT or LOAD TABLE statement, and insert into the other columns in the same row with additional INSERT or LOAD TABLE statements.

If you try to insert into a column that already contains data, you get an error.

You must be sure to control the row at which each insertion starts. If you do not use START ROW ID, your insertion begins with the next row, and NULLs are inserted in the remaining columns of the current row, as shown in Figure 7-1. (The two shading patterns represent data inserted into columns in two separate insert operations.)

**Figure 7-1: Using START ROW ID with partial-width insertions**



---

**Note** Do not try to perform a partial-width insertion using the `INSERT VALUES` command format. Because you cannot specify START ROW ID using this format, the problem shown in the figure results.

---

## Partial-width insertion rules

Column indexes that are not included in the initial partial-width insert, and therefore do not already contain data, must allow NULLs. Sybase IQ inserts NULLs into these column indexes. If they do not allow NULLs, the insert fails.

When doing partial-width inserts, follow these steps:

- 1 For the first partial-width insert for each set of rows, do not specify START ROW ID. Sybase IQ automatically knows what the next available row is for this insert.
- 2 For the second and any subsequent partial-width inserts for the same set of rows, use the START ROW ID option to specify the row where the insert started. This number is the record number at the beginning of the insert message log, as in this example:

```
In table 'department', the full width insert of 3
columns
will begin at record 1.
```

You can also use the ROWID function to display the row ID, as in the following query:

```
SELECT *, ROWID(table_name) FROM table_name
```

#### Example 1

The UNIX example below shows an incorrect insertion of four columns from the file *tt.t* into the indexes on the lineitem table. It inserts the first two columns with one LOAD TABLE statement and the second two columns with another LOAD TABLE statement, but does not use the START ROW ID option to align the additional columns.

```
LOAD TABLE lineitem
  (l_partkey ASCII(4),
   l_suppkey ASCII(4),
   FILLER(13))
FROM '/d1/MILL1/tt.t'
PREVIEW ON
NOTIFY 1000
```

```
LOAD TABLE lineitem
  (FILLER(8),
   l_quantity ASCII(6),
   l_orderkey ASCII(6),
   FILLER(1))
FROM '/d1/MILL1/tt.t'
PREVIEW ON
NOTIFY 1000
```

The result of the SELECT statement below shows that 10 rows are stored instead of the correct number of 5.

```
SELECT *, rowid(lineitem) FROM lineitem
```

<i>l_orderkey</i>	<i>l_partkey</i>	<i>l_suppkey</i>	<i>l_quantity</i>	rowid( <i>lineitem</i> )
NULL	1	12	NULL	1
NULL	2	37	NULL	2
NULL	3	28	NULL	3
NULL	4	13	NULL	4
NULL	5	9	NULL	5
190	NULL	NULL	19	6
215	NULL	NULL	2127	7
29	NULL	NULL	1376	8
200	NULL	NULL	119	9
59	NULL	NULL	4	10

(10 rows affected)

**Example 2**

The following example shows the correct way to do this operation. Note the START ROW ID option in the second LOAD TABLE statement.

```
LOAD TABLE lineitem
  (l_partkey ASCII(4),
   l_suppkey ASCII(4),
   FILLER(13))
  FROM '/d1/MILL1/tt.t'
  PREVIEW ON
  NOTIFY 1000

  SELECT *, rowid(lineitem) FROM lineitem
l_orderkey  l_partkey  l_suppkey  l_quantity  rowid(lineitem)
-----  -----  -----  -----  -----
NULL        1          12         NULL        1
NULL        2          37         NULL        2
NULL        3          28         NULL        3
NULL        4          13         NULL        4
NULL        5          9          NULL        5

(5 rows affected)

LOAD TABLE lineitem
  (FILLER(8),
   l_quantity ASCII(6),
   l_orderkey ASCII(6),
   FILLER(1))
  FROM '/d1/MILL1/tt.t'
  PREVIEW ON
  NOTIFY 1000
  START ROW ID 1

  SELECT *, rowid(lineitem) FROM lineitem
l_orderkey  l_partkey  l_suppkey  l_quantity  rowid(lineitem)
-----  -----  -----  -----  -----
190         1          12          19          1
215         2          37          2127        2
29          3          28          1376        3
200         4          13          119         4
59          5          9           4          5

(5 rows affected)
```

To ensure that the data from the second two columns is inserted into the same rows as the first two columns, you must specify the row number in the START ROW ID option on the INSERT command for the next two columns.

**Using the FILLER Option**

The FILLER option tells Sybase IQ which columns in the input file to skip. This LOAD TABLE statement inserts NULLs into the second two columns, because those columns are skipped. Note that these columns must allow NULLs in order for this statement to work.

**Example 3**

For this next Windows example, assume the partsupp table has two columns, ps\_partkey and ps\_availqty, and that partsupp is not part of any join index.

The data for ps\_value is calculated from ps\_availqty so the ps\_availqty column must already contain data. Therefore, to insert data into the partsupp table, do two inserts: one for ps\_availqty and ps\_partkey and then one for ps\_value.

First, insert the data for partsupp directly from an ASCII file named *tt.t*.

```
LOAD TABLE partsupp
      (ps_partkey ASCII(6),
       ps_availqty ASCII(6),
       FILLER(2))
FROM 'C:\\\\iq\\\\archive\\\\mill1.txt'

SELECT *, rowid(partsupp) FROM partsupp
ps_partkey  ps_suppkey    ps_availqty   ps_value   rowid(partsupp)
-----  -----  -----  -----
213        NULL          190           NULL        1
24         NULL          215           NULL        2

(2 rows affected)
```

Next select the ps\_availqty and do an 80% calculation. In this case you must use an INSERT command to insert the results of a SELECT statement.

```
INSERT INTO partsupp(ps_value)
START ROW ID 1
SELECT ps_availqty * 0.80 FROM partsupp

SELECT *, rowid(partsupp) FROM partsupp
ps_partkey  ps_suppkey    ps_availqty   ps_value   rowid(partsupp)
-----  -----  -----  -----
213        NULL          190          152.00      1
24         NULL          215          172.00      2

(2 rows affected)
```

If you later load data from another file into ps\_partkey and ps\_availqty, insertions begin correctly at the next row, as shown below.

```
LOAD TABLE partsupp
  (ps_partkey ASCII(6),
   ps_availqty ASCII(6),
   FILLER(2))
FROM 'C:\\iq\\archive\\mill2.txt'

      SELECT *, rowid(partsupp) FROM partsupp
ps_partkey  ps_suppkey    ps_availqty  ps_value  rowid(partsupp)
-----  -----  -----  -----
213        NULL          190          152.00     1
24         NULL          215          172.00     2
28         NULL          490          NULL       3
211        NULL          15           NULL       4

(4 rows affected)
```

To calculate and insert the values for ps\_value, you need to repeat the INSERT statement shown earlier in this example, changing the START ROW ID value to the new row number, 3.

Previewing partial-width inserts

Given the possibility of errors if you do a partial-width insert incorrectly, it is a good idea to preview these inserts. The PREVIEW load option lets you see the layout of input in the destination table. This option is available in LOAD TABLE, but not in the INSERT command.

## Converting data 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 CHAR data into an INT 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.

For information on implicit and explicit conversions between Sybase IQ data types, see the tables in the section “Data conversions in IQ” on page 367.

For information on conversions that occur if you are inserting from proxy tables, see Chapter 16, “Accessing Remote Data.”.

While most Sybase IQ data types are fully compatible with Adaptive Server Anywhere and Adaptive Server Enterprise data types of the same name, there are some differences. For details on compatibility, see “Matching Adaptive Server Enterprise data types” on page 382.

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 Adaptive Server 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, Adaptive Server IQ loads only the first 2 bytes of that input column.

---

## Inserting data from pre-Version 12 Sybase IQ

If you are moving data into Sybase IQ Version 12.x from an earlier version, you must convert certain data types before inserting or loading them. For details, see “Migrating Data from Prior Versions” in the *Sybase IQ 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. For a detailed description of each option, see the sections that follow. To use these options in the LOAD TABLE statement, see “Specifying load options”.

**Table 7-5: 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.

Option	Sybase IQ Data types	Action
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.
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.

## 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 an Sybase IQ database, or coming from an Adaptive Server 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 7-6: IQ conversions for comparison operations**

From:	To:																	
	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	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	E	E	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	
timestamp	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	E	E	E	E	I	I	I	I	
varchar	E	E	E	E	E	E	E	E	E	E	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 Table 7-6, Table 7-7, and Table 7-8:

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 7-7: IQ conversions for arithmetic operations**

<b>From:</b>	<b>To:</b>																	
	<b>ti</b>	<b>si</b>	<b>in</b>	<b>ui</b>	<b>bi</b>	<b>ub</b>	<b>nu</b>	<b>rl</b>	<b>dl</b>	<b>bt</b>	<b>dt</b>	<b>tm</b>	<b>ts</b>	<b>ch</b>	<b>vc</b>	<b>bn</b>	<b>vb</b>	
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	E	E	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	
timestamp	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	E	U	U	U	I	I	I	I	
varchar	E	E	E	E	E	E	E	E	E	E	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 7-8: IQ conversions for INSERT and UPDATE**

<b>From:</b>	<b>To:</b>																
	<b>ti</b>	<b>si</b>	<b>in</b>	<b>ui</b>	<b>bi</b>	<b>ub</b>	<b>nu</b>	<b>rl</b>	<b>dl</b>	<b>bt</b>	<b>dt</b>	<b>tm</b>	<b>ts</b>	<b>ch</b>	<b>vc</b>	<b>bn</b>	<b>vb</b>
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	E	E	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	E	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	E	E	E	E	I	I	I	I
varchar	I	I	I	I	I	I	I	I	I	E	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

## 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.

## Optimizing 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"
- "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 foo(c1 DATE);
```

To load the table, use a statement like this:

```
LOAD TABLE foo (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.

## Using the ASCII conversion option

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 7-9: Input file conversion example**

<i>shipinfo.t</i>			<i>lineitem</i>		
<i>column</i>	<i>format</i>	<i>width</i>	<i>column</i>	<i>datatype</i>	<i>width</i>
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 the *Sybase IQ Reference Manual*.

## 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 NULLS (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

```

## Specifying the DATE Format

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 7-10 describes the formatting options.

**Table 7-10: Formatting dates**

Option	Meaning
yyyy or YYYY	Represents number of year. Default is 1900.
yy or YY	
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	Represents number of day. Default day is 01.
jjj or JJJ	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 of 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 zeroes.
- Same case (for example, “DD” or “dd” means do pad with zeroes.

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 7-11: Sample DATE format options**

<b>Input Data</b>	<b>Format Specification</b>
12/31/00	DATE ('MM/DD/YY')
12-31-00	DATE ('MM-DD-YY')
20001231	DATE ('YYYYMMDD')
12/00	DATE ('MM/YY')
2000/123	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, 2001-33 is the 33rd day in the year 2001, or February 2, 2001.
- 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.

<b>NEAREST_CENTURY setting</b>	<b>Year specified as</b>	<b>Years assumed</b>
Default (50)	00-49	2000-2049
	50-99	1950-1999
0	any	1900s
100	any	2000s

For more information, see Chapter 2, “Database Options,” in the *Sybase IQ Reference Manual*.

## 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'
BLOCK FACTOR 1000
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'
BLOCK FACTOR 1000
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 customer(
    open_time DATETIME('hh:mm:ss'),
```

```
close_time DATETIME('hh:mm:ss') ,
FILLER(9)
FROM '/d1/MILL1/tt.t'
BLOCK FACTOR 1000
PREVIEW ON
```

## **Specifying the format for DATETIME conversions**

Specify the format of the DATETIME input data 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. Table 7-10 describes the date formatting options. The following table describes the time formatting options.

**Table 7-12: Formatting times**

Option	Meaning
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.
HH	Represents minute. Always use leading zeros for minute where appropriate, for example '08' for 8 minutes.
nn	Represents seconds and fraction of a second.
ss[.ssssss]	Represents the a.m. or p.m designation.
aa	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.)
pp	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	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'.
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 Table 7-13 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 7-13: DATETIME format options**

Input Data	Format Specification
12/31/00 14:01:50	DATETIME ('MM/DD/YY hh:nn:ss')
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:ssssss')
12/31/00 14:01:.123456	DATETIME ('MM/DD/YY hh:mm:ssssss')
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.

## Working with NULLS

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.

Here is the syntax.

`NULLS ({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.

Here is a Windows example:

```
LOAD TABLE lineitem(
    l_orderkey NULLS (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. See Chapter 5, “SQL Functions,” in the *Sybase IQ Reference Manual* for more information.
- 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 the *Sybase IQ Reference Manual*.

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.

Ensuring accurate dates

For dates in the first 9 days of a month, Adaptive Server Enterprise supports a blank or zero for the first digit; IQ supports a zero and a blank.

To ensure accuracy in dates, follow the recommendation in the *Adaptive Server Enterprise Reference Manual* to use leading zeros for single-digit years, months, and days.

If your Adaptive Server Enterprise dates include a blank first digit for the date, and you must use bcp out to export such data,

## Unsupported Adaptive Server Enterprise data types

These Adaptive Server Enterprise data types are not supported by Sybase IQ in this version:

- nchar, nvarchar
- unichar, univarchar
- text
- image

- unsigned smallint
- long varchar
- native Java data types

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 *Large Objects Management in Sybase IQ*.
- In Sybase Central, the Table Editor allows you to create columns with certain datatypes in the SYSTEM dbspace (Catalog Store) only. Choose the Advanced Table Properties button (fourth from the left) on the toolbar, then select SYSTEM from the DB Space dropdown box. The Data Type dropdown will now include datatypes such as TEXT and java.lang.Object. Selecting SYSTEM also enables Include/Exclude Java Class Datatypes, the third icon from the right in the Table Editor toolbar. Choose this icon to add these datatypes to the Data Type dropdown.
- Sybase IQ does not support the Adaptive Server Enterprise data type TEXT, but you can insert data from an ASE database column of data type TEXT 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 7-14: Integer data types**

<b>Adaptive Server Enterprise Datatype</b>	<b>Sybase IQ Datatype</b>	<b>Notes</b>
int	INT,BIGINT,UNSIGNED INT, UNSIGNED BIGINT, or NUMERIC	<p>Sybase IQ does not allow scaled integers, such as INT(7,3). Data in the form INT(<i>precision,scale</i>) is converted to NUMERIC(<i>precision,scale</i>). 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.</p> <p>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.</p>
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.
smallint	SMALLINT or NUMERIC	Sybase IQ SMALLINT does not allow precision and scale. Adaptive Server Enterprise smallint( <i>precision,scale</i> ) is converted to NUMERIC( <i>precision,scale</i> ). See INT above.
tinyint	TINYINT	Sybase IQ TINYINT columns do not allow precision and scale. Adaptive Server Enterprise tinyint( <i>precision,scale</i> ) is converted to NUMERIC( <i>precision,scale</i> ). See INT above.
bit	BIT	

The following table indicates the Adaptive Server Enterprise approximate data types and the Sybase IQ equivalents.

**Table 7-15: Approximate numeric data types**

<b>Adaptive Server Enterprise Datatype</b>	<b>Sybase IQ Datatype</b>	<b>Notes</b>
float ( <i>precision</i> )	FLOAT ( <i>precision</i> )	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 7-16: Character data types**

<b>Adaptive Server Enterprise Datatype</b>	<b>Sybase IQ Datatype</b>	<b>Notes</b>
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	
nvarchar	Not supported	
text	Not supported	Sybase IQ does not support the Adaptive Server Enterprise data type text, but you can insert data from an ASE database column of data type text using INSERT...LOCATION.

The following table indicates the Adaptive Server Enterprise money data types and the Sybase IQ equivalents.

**Table 7-17: Money data types**

<b>Adaptive Server Enterprise Datatype</b>	<b>Sybase IQ Datatype</b>	<b>Notes</b>
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 7-18: DATE/TIME data types**

<b>Adaptive Server Enterprise Datatype</b>	<b>Sybase IQ Datatype</b>	<b>Notes</b>
datetime	TIMESTAMP or DATE or TIME	<p>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.</p> <p>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.</p>
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.

The following table indicates the Adaptive Server Enterprise binary data types and the Sybase IQ equivalents.

**Table 7-19: Binary data types**

<b>Adaptive Server Enterprise Datatype</b>	<b>Sybase IQ Datatype</b>	<b>Notes</b>
binary	BINARY	<p>Sybase IQ pads trailing zeros on all BINARY columns. Always create BINARY columns with an even number of characters for length.</p> <p>HNG indexes do not allow BINARY data.</p>
varbinary	VARBINARY	<p>Sybase IQ does not pad or truncate trailing zeros on VARBINARY columns. Always create VARBINARY columns with an even number of characters for length.</p> <p>HNG indexes do not allow VARBINARY data.</p> <p>If you use INSERT ... LOCATION to insert data selected from a VARBINARY column, set the LOAD_MEMORY_MB option on the <i>local</i> database to limit memory used by the insert, and set ASE_BINARY_DISPLAY to 'OFF' on the <i>remote</i> database.</p>

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.

## Handling conversion errors on data import

When you are loading 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 “SET OPTION statement” in the *Sybase IQ Reference Manual*.

## Tuning 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.

## Improving load performance during database definition

The way you define your database, tables, and indexes can have a dramatic impact on load performance.

## Optimizing for the 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.

For details of how these parameters affect loading, and information on how to specify them, see “Creating tables” on page 239 and “Choosing an IQ page size” on page 189.

## Creating indexes

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.

## Adding dbspaces

If you run out of space while loading data, Sybase IQ prompts you to create another dbspace, and then continues the operation after you add the dbspace. To avoid this delay, make sure that you have enough room for all of the data you are loading before you start the load operation. Use the sp\_estspace or sp\_iqestdbspaces stored procedure to help you estimate the space you need for the database and its dbspaces.

To ensure you are able to add a new dbspace if you do run out of space, see the “TEMP\_RESERVED\_DBSPACE\_MB option” and “MAIN\_RESERVED\_DBSPACE\_MB option” options in the *Sybase IQ Reference Manual*.

## Setting server startup options

On some platforms you can set command-line options to adjust the amount of memory available. Increasing memory can improve load performance. See Chapter 4, “Connection and Communication Parameters,” for command-line options that affect performance.

## Adjusting your environment at load time

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, as described in “Bulk loading data using the LOAD TABLE statement.” In particular, if you have sufficient memory to do so, or if no other users are active during the load, increase the BLOCK FACTOR.

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 command line options iqmc and iqtc to increase the cache size; see “Server command-line switches” in Chapter 1, *Sybase IQ Utility Guide* for details. For these options to take effect, you must restart the server.
- Adjust the amount of heap memory used by load operations by using the SET OPTION command to change the LOAD\_MEMORY\_MB option. When LOAD\_MEMORY\_MB is set to the default (0), Sybase IQ uses the amount of heap memory that gives the best performance. If your system runs out of virtual memory, specify a value less than 2000 and decrease the value until the load works. For insertions into wide tables, you may need to set LOAD\_MEMORY\_MB to a low value (100-200 MB). If you set the value too low, it may be physically impossible to load the data. Note that this option also affects INSERT, UPDATE, SYNCHRONIZE, and DELETE operations.
- If you are loading very wide varchar data, reduce the value of the LOAD\_MEMORY\_MB database option and the BLOCK FACTOR option in the LOAD TABLE command. As with all performance tuning, adjusting these values may require experimentation.

- 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. See “Managing lock contention” on page 489 for more information.
- 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 DBISQL 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. For details on the syntax of the `INSERT` statement, see Chapter 6, “SQL Statements,” chapter of the *Sybase IQ Reference Manual*.

## Reducing Main IQ 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.

For more information on versioning see Chapter 10, “Transactions and Versioning”.

## Understanding 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, and partial-width variable-width loads, 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.

## Changing data using UPDATE

You can use the UPDATE statement, followed by the name of the table or view, to change single rows, groups of rows, or all rows in a table. As in all data modification statements, you can change the data in only one table or view at a time.

The UPDATE statement specifies the row or rows you want changed and the new data. The new data can be a constant or an expression that you specify or data pulled from other tables.

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. See Table 7-8 on page 370 in the section “Data conversions in IQ” for details on data type conversions for the UPDATE statement.

### UPDATE syntax

See the *Sybase IQ Reference Manual* for complete UPDATE syntax. A simplified version of the syntax is:

**UPDATE** *table-name*

**SET** *column\_name* = *expression*

**WHERE** *search-condition*

If the company Newton Ent. (in the customer table of the sample database) is taken over by Einstein, Inc., you can update the name of the company using a statement such as the following:

```
UPDATE customer  
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 customer  
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 customer  
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 product  
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.

## Deleting 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.

For syntax of these statements, see the *Sybase IQ Reference Manual*.

### 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. For details, see “Overlapping versions and deletions”.

When you use `DROP TABLE`, you do not need to add space, as no extra version pages are needed.



# Using Procedures and Batches

## About this chapter

This chapter explains how you create procedures and batches for use with Sybase IQ.

Procedures store procedural SQL statements in the database for use by all applications. They enhance the security, efficiency, and standardization of databases. User-defined functions are one kind of procedure that return a value to the calling environment for use in queries and other SQL statements. Batches are sets of SQL statements submitted to the database server as a group. Many features available in procedures, such as control statements, are also available in batches.

For many purposes, server-side JDBC provides a more flexible way to build logic into the database than SQL stored procedures. For information on JDBC, see Appendix B, “Data Access Using JDBC”.

## Contents

Topic	Page
Overview of procedures	396
Benefits of procedures	396
Introduction to procedures	397
Introduction to user-defined functions	404
Introduction to batches	407
Control statements	408
Structure of procedures	411
Returning results from procedures	414
Using cursors in procedures	420
Errors and warnings in procedures	424
Using the EXECUTE IMMEDIATE statement in procedures	434
Transactions and savepoints in procedures	434
Tips for writing procedures	434
Statements allowed in batches	437
Using IQ UTILITIES to create your own stored procedures	438

## Overview of procedures

Procedures store procedural SQL statements in a database for use by all applications. They can include control statements that allow repetition (LOOP statement) and conditional execution (IF statement and CASE statement) of SQL statements.

Procedures are invoked with a CALL statement, and use parameters to accept values and return values to the calling environment. Procedures can also return result sets to the caller, or call other procedures.

User-defined functions are one kind of stored procedure that returns a single value to the calling environment. User-defined functions do not modify parameters passed to them, but rather, broaden the scope of functions available to queries and other SQL statements.

### Procedure debugger

You can debug stored procedures using the combined stored procedure/Java debugger. For more information, see Appendix C, “Debugging Logic in the Database.”

## Benefits of procedures

Definitions for procedures appear in the database, separately from any one database application. This separation provides a number of advantages.

### Standardization

Procedures standardize actions performed by more than one application program. By coding the action once and storing it in the database for future use, applications need only call the procedure to achieve the desired result repeatedly. And since changes occur in only one place, all applications using the action automatically acquire the new functionality if the implementation of the action changes.

### Efficiency

Procedures used in a network database server environment can access data in the database without requiring network communication. This means they execute faster and with less impact on network performance than if they had been implemented in an application on one of the client machines.

When you create a procedure, it is automatically checked for correct syntax, and then stored in the system tables. The first time any application calls a procedure, it is compiled from the system tables into the virtual memory of the server and executed from there. Since one copy of the procedure remains in memory after the first execution, repeated executions of the same procedure happen instantly. As well, several applications can use a procedure concurrently, or one application can use it recursively.

#### Security

Procedures provide security by allowing users limited access to data in tables that they cannot directly examine or modify.

Procedures, including user-defined functions, execute with the permissions of the procedure owner but can be called by any user that has been granted permission to do so.

This means that a procedure can (and usually does) have different permissions than the user ID that invoked it. Procedures provide security by allowing users limited access to data in tables that they cannot directly examine or modify.

## Introduction to procedures

In order to use procedures, you need to understand how to:

- Create procedures
- Call procedures from a database application
- Drop or remove procedures
- Control who has permission to use procedures

This section discusses these aspects of using procedures, and also describes some of the different uses of procedures.

Two system stored procedures that are useful when working with stored procedures are `sp_iqprocedure` and `sp_iqprocparm`. The `sp_iqprocedure` stored procedure displays information about system and user-defined procedures in a database. The `sp_iqprocparm` stored procedure displays information about stored procedure parameters, including result set variables and SQLSTATE/SQLCODE error values.

## Creating procedures

Procedures are created using the CREATE PROCEDURE statement. You must have RESOURCE authority in order to create a procedure.

Where you enter the statement depends on the tool you are using.

### ❖ Creating a new procedure (Sybase Central)

- 1 Connect to a database with DBA or Resource authority.
- 2 Open the Procedures folder of the database.
- 3 In the right pane, double-click Add Procedure (Wizard).
- 4 Follow the instructions in the wizard.
- 5 When the Code Editor opens, complete the code of the procedure.
- 6 To execute the code in the database, choose File > Save/Execute in Database.

The new procedure appears in the Procedures folder.

### ❖ Creating a procedure (SQL)

- 1 Launch Interactive SQL and connect to a database using DBA authority.
- 2 Type the commands for the procedure in the SQL Statements pane of the Interactive SQL viewer.

If you are using a tool other than Interactive SQL or Sybase Central, follow the instructions for your tool. You may need to change the command delimiter away from the semicolon before entering the CREATE PROCEDURE statement.

---

**Note** To create a remote procedure in IQ, you must use the AT *location-string* SQL syntax of CREATE PROCEDURE to create a proxy stored procedure. This capability is currently certified on Windows and Sun Solaris only. For more information, see “Using remote procedure calls (RPCs)” on page 711. The Add Remote Procedure Wizard in Sybase Central does not support remote IQ procedures.

---

### Example

The following simple example creates the procedure new\_dept, which carries out an INSERT into the department table of the sample database, creating a new department.

```
CREATE PROCEDURE new_dept ( IN id INT,  
                           IN name CHAR(35) ,
```

```
    IN head_id INT )
BEGIN
    INSERT
        INTO DBA.department ( dept_id,
        dept_name,
        dept_head_id )
    VALUES ( id, name, head_id );
END
```

For a complete description of the CREATE PROCEDURE syntax, see Chapter 6, “SQL Statements,” of the *Sybase IQ Reference Manual*.

The body of a procedure is a **compound statement**. The compound statement starts with a BEGIN statement and concludes with an END statement. In the case of new\_dept, the compound statement is a single INSERT bracketed by BEGIN and END statements.

For more information, see “Using compound statements” on page 409.

Parameters to procedures are marked as one of IN, OUT, or INOUT. All parameters to the new\_dept procedure are IN parameters, as they are not changed by the procedure.

## Altering procedures

You can modify an existing procedure using either Sybase Central or Interactive SQL. You must have DBA authority or be the owner of the procedure.

In Sybase Central, you cannot rename an existing procedure directly. Instead, you must create a new procedure with the new name, copy the previous code to it, and then delete the old procedure.

In Interactive SQL, you can use an ALTER PROCEDURE statement to modify an existing procedure. You must include the entire new procedure in this statement (in the same syntax as in the CREATE PROCEDURE statement that created the procedure). You must also reassign user permissions on the procedure.

For information on altering database object properties, see in *Introduction to Sybase IQ*, Chapter 4, “Managing Databases.”

For information on granting or revoking permissions for procedures, see “Granting permissions on procedures” on page 572 and “Revoking user permissions” on page 573.

❖ **Altering the code of a procedure (Sybase Central)**

- 1 Open the Procedures & Functions folder.
- 2 Right-click the desired procedure.
- 3 From the popup menu, do one of the following:
  - Choose Open as Watcom-SQL to edit the code in the Watcom-SQL dialect.
  - Choose Open as Transact-SQL to edit the code in the Transact-SQL dialect.
- 4 In the Code Editor, edit the procedure's code.
- 5 To execute the code in the database, choose File > Save/Execute in Database.

❖ **Altering the code of a procedure (SQL)**

- 1 Connect to the database.
- 2 Execute an ALTER PROCEDURE statement. Include the entire new procedure in this statement.

For more information, see “Creating procedures” on page 398, and see “ALTER PROCEDURE statement” and “CREATE PROCEDURE statement” in the *Sybase IQ Reference Manual*.

## Calling procedures

CALL statements invoke procedures. Procedures can be called by an application program or by other procedures.

For more information, see “CALL statement” in Chapter 6, “SQL Statements,” of the *Sybase IQ Reference Manual*.

The following statement calls the new\_dept procedure to insert an Eastern Sales department:

```
CALL new_dept( 210, 'Eastern Sales', 902 );
```

After this call, you may wish to check the department table to see that the new department has been added.

All users who have been granted EXECUTE permissions for the procedure can call the new\_dept procedure, even if they have no permissions on the department table.

For more information about EXECUTE permissions, see “Permissions to execute procedures” on page 402.

## Copying procedures in Sybase Central

In Sybase Central, you can copy procedures between databases. To do so, select the procedures in the right pane of Sybase Central and drag it to the Procedures & Functions folder of another connected database. A new procedure is then created, and the original procedure's code is copied to it.

Note that only the procedure code is copied to the new procedure. The other procedure properties (permissions, etc.) are not copied. A procedure can be copied to the same database, provided it is given a new name.

## Deleting procedures

Once you create a procedure, it remains in the database until someone explicitly removes it. Only the owner of the procedure or a user with DBA authority can drop the procedure from the database.

### ❖ Deleting a procedure (Sybase Central)

- 1 Connect to a database with DBA authority or as the owner of the procedure.
- 2 Open the Procedures & Functions folder.
- 3 Right-click the desired procedure and choose Delete from the popup menu.

### ❖ Deleting a procedure (SQL)

- 1 Connect to a database with DBA authority or as the owner of the procedure.
- 2 Execute a DROP PROCEDURE statement.

### Example

The following statement removes the procedure new\_dept from the database:

```
DROP PROCEDURE new_dept
```

## Permissions to execute procedures

A procedure is owned by the user who created it, and that user can execute it without permission. Permission to execute the procedure can be granted to other users using the GRANT EXECUTE command.

For example, the owner of the procedure new\_dept allows another\_user to execute new\_dept with the statement:

```
GRANT EXECUTE ON new_dept TO another_user
```

The following statement revokes permission to execute the procedure:

```
REVOKE EXECUTE ON new_dept FROM another_user
```

For more information on managing user permissions on procedures, see “Granting permissions on procedures” on page 572.

## Returning procedure results in parameters

Procedures return results to the calling environment in one of the following ways:

- Individual values are returned as OUT or INOUT parameters.
- Result sets can be returned.
- A single result can be returned using a RETURN statement.

This section describes how to return results from procedures as parameters.

The following procedure on the sample database returns the average salary of employees as an OUT parameter.

```
CREATE PROCEDURE AverageSalary( OUT avgsal
                                NUMERIC (20,3) )
BEGIN
    SELECT AVG( salary )
    INTO avgsal
    FROM employee;
END
```

### ❖ Running this procedure and displaying its output (SQL)

- 1 Connect to the sample database from Interactive SQL with a user ID of DBA and a password of SQL.
- 2 In the SQL Statements pane, type the above procedure code.

- 3 Create a variable to hold the procedure output. In this case, the output variable is numeric, with three decimal places, so create a variable as follows:

```
CREATE VARIABLE Average NUMERIC(20, 3)
```

- 4 Call the procedure using the created variable to hold the result:

```
CALL AverageSalary(Average)
```

If the procedure was created and run properly, the Interactive SQL Messages pane does not display any errors.

- 5 Execute the SELECT Average statement to inspect the value of the variable.

Look at the value of the output variable Average. The Interactive SQL Results pane displays the value 49988.623 for this variable, the average employee salary.

## Returning procedure results in result sets

In addition to returning results to the calling environment in individual parameters, procedures can return information in result sets. A result set is typically the result of a query. The following procedure returns a result set containing the salary for each employee in a given department:

```
CREATE PROCEDURE SalaryList (IN department_id INT)
RESULT ( "Employee ID" INT, "Salary" NUMERIC(20, 3) )
BEGIN
    SELECT emp_id, salary
    FROM employee
    WHERE employee.dept_id = department_id;
END
```

If Interactive SQL calls this procedure, the names in the RESULT clause are matched to the results of the query and used as column headings in the displayed results.

To test this procedure from Interactive SQL, you can call it, specifying one of the departments of the company in the CALL statement. The results appear in the Interactive SQL Results pane.

### Example

To list the salaries of employees in the R & D department (department ID 100), type the following:

```
CALL SalaryList (100)
```

Employee ID	Salary
102	45700.000
105	62000.000
160	57490.000
243	72995.000
247	48023.690

Interactive SQL can only return multiple result sets if you have this option enabled on the Commands tab of the Options dialog. For more information, see “Returning multiple result sets from procedures” on page 418.

Creating and selecting from temporary tables

If a procedure dynamically creates and then selects the same temporary table within a stored procedure, you must use the EXECUTE IMMEDIATE WITH RESULT SET ON syntax to avoid “Column not found” errors.

For example:

```
CREATE PROCEDURE p1 (IN @t varchar(30))
BEGIN
    EXECUTE IMMEDIATE
        'SELECT * INTO #resultSet FROM ' || @t;
    EXECUTE IMMEDIATE WITH RESULT SET ON
        'SELECT * FROM #resultSet';
END
```

## Introduction to user-defined functions

User-defined functions are a class of procedures that return a single value to the calling environment. This section introduces creating, using, and dropping user-defined functions.

### Creating user-defined functions

You use the CREATE FUNCTION statement to create user-defined functions. However, you must have RESOURCE authority.

The following simple example creates a function that concatenates two strings, together with a space, to form a full name from a first name and a last name.

```
CREATE FUNCTION fullname (firstname CHAR(30),
```

```
lastname CHAR(30))
RETURNS CHAR(61)
BEGIN
    DECLARE name CHAR(61);
    SET name = firstname || ' ' || lastname;
    RETURN ( name );
END
```

❖ **Creating this example using Interactive SQL**

- 1 Connect to the sample database from Interactive SQL with a user ID of DBA and a password of SQL.
- 2 In the SQL Statements pane, type the above function code.

---

**Note** If you are using a tool other than Interactive SQL or Sybase Central, you may need to change the command delimiter away from the semicolon before entering the CREATE FUNCTION statement.

---

For a complete description of the CREATE FUNCTION syntax, including performance considerations and differences between Adaptive Server Anywhere and IQ, see Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

The CREATE FUNCTION syntax differs slightly from that of the CREATE PROCEDURE statement. The following are distinctive differences:

- No IN, OUT, or INOUT keywords are required, as all parameters are IN parameters.
- The RETURNS clause is required to specify the data type being returned.
- The RETURN statement is required to specify the value being returned.

## Calling user-defined functions

A user-defined function can be used, subject to permissions, in any place you would use a built-in non-aggregate function.

The following statement in Interactive SQL returns a full name from two columns containing a first and last name:

```
SELECT fullname (emp_fname, emp_lname)
FROM employee;
```

**fullname (emp\_fname, emp\_lname)**

---

Fran Whitney

---

Matthew Cobb

---

Philip Chin

---

...

The following statement in dbisql returns a full name from a supplied first and last name:

```
SELECT fullname ('Jane', 'Smith');
```

**fullname ('Jane','Smith')**

---

Jane Smith

Any user who has been granted EXECUTE permissions for the function can use the fullname function.

## Dropping user-defined functions

Once a user-defined function is created, it remains in the database until it is explicitly removed. Only the owner of the function or a user with DBA authority can drop a function from the database.

The following statement removes the function fullname from the database:

```
DROP FUNCTION fullname
```

## Permissions to execute user-defined functions

A user-defined function is owned by the user who created it, and that user can execute it without permission. The owner of a user-defined function can grant permissions to other users with the GRANT EXECUTE command.

For example, the creator of the function fullname allows another\_user to use fullname with the statement:

```
GRANT EXECUTE ON fullname TO another_user
```

The following statement revokes permission to use the function:

```
REVOKE EXECUTE ON fullname FROM another_user
```

For more information on managing user permissions on functions, see “Granting permissions on procedures” on page 572.

## Introduction to batches

A simple batch consists of a set of SQL statements, separated by semicolons. For example, the following statements form a batch that creates an Eastern Sales department and transfers all sales representatives from Massachusetts (MA) to that department.

```
INSERT
  INTO department ( dept_id, dept_name )
    VALUES ( 220, 'Eastern Sales' ) ;

UPDATE employee
  SET dept_id = 220
 WHERE dept_id = 200
   AND state = 'MA' ;

COMMIT ;
```

You can include this set of statements in an application and execute them together.

---

### ***dbisql and batches***

A list of semicolon-separated statements, such as the above, is parsed by dbisql before it is sent to the server. In this case, dbisql sends each statement individually to the server, not as a batch. Unless you have such parsing code in your application, the statements are sent and treated as a batch. Putting a BEGIN and END around a set of statements causes dbisql to treat them as a batch.

---

Many statements used in procedures can also be used in batches. You can use control statements (CASE, IF, LOOP, and so on), including compound statements (BEGIN and END), in batches. Compound statements can include declarations of variables, exceptions, temporary tables, or cursors inside the compound statement.

The following batch creates a table only if a table of that name does not already exist:

```
BEGIN
  IF NOT EXISTS (
    SELECT * FROM SYSTABLE
```

```
        WHERE table_name = 't1' ) THEN
        CREATE TABLE t1 (
                firstcol INT PRIMARY KEY,
                secondcol CHAR( 30 )
        ) ;
    ELSE
        MESSAGE 'Table t1 already exists' ;
    END IF
END
```

If you run this batch twice from dbisql, it creates the table the first time you run it and displays the message in the dbisql messages window the next time you run it.

## Control statements

There are a number of control statements for logical flow and decision making in the body of the procedure or in a batch. Available control statements include:

Control statement	Syntax
Compound statements	BEGIN [ ATOMIC ] statement-list END
Conditional execution: IF	IF condition THEN statement-list ELSEIF condition THEN statement-list ELSE statement-list END IF
Conditional execution: CASE	CASE expression WHEN value THEN statement-list WHEN value THEN statement-list ELSE statement-list END CASE
Repetition: WHILE, LOOP	WHILE condition LOOP statement-list END LOOP

Control statement	Syntax
Repetition: FOR cursor loop	FOR loop-name AS cursor-name CURSOR FOR select statement DO Statement-list END FOR
Break: LEAVE	LEAVE label
CALL	CALL procname( arg, ... )

For complete descriptions of each, see the entries in Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

## Using compound statements

A compound statement starts with the keyword BEGIN and ends with the keyword END. The body of a procedure is a **compound statement**. Compound statements can also be used in batches. Compound statements can be nested, and combined with other control statements to define execution flow in procedures or in batches.

A compound statement allows a set of SQL statements to be grouped together and treated as a unit. SQL statements within a compound statement should be separated with semicolons.

A command delimiter is required after every statement in a statement list except for the last, where it is optional.

## Declarations in compound statements

Local declarations in a compound statement immediately follow the BEGIN keyword. These local declarations exist only within the compound statement. The following may be declared within a compound statement:

- Variables
- Cursors
- Temporary tables
- Exceptions (error identifiers)

Local declarations can be referenced by any statement in that compound statement, or in any compound statement nested within it. Local declarations are not visible to other procedures called from the compound statement.

## Atomic compound statements

An **atomic** statement is a statement that is executed completely or not at all. For example, a LOAD statement that inserts thousands of rows might encounter an error after many rows. If the statement does not complete, and the default ON FILE ERROR ROLLBACK option is in effect, all changes are undone. This LOAD statement is atomic.

All noncompound SQL statements are atomic. A compound statement can be made atomic by adding the keyword ATOMIC after the BEGIN keyword.

```
BEGIN ATOMIC
  INSERT INTO
    sales_order (id, order_date, sales_rep)
    VALUES (41880, 1998-08-24, 2054) ;

  INSERT INTO
    sales_order_items (line_id, prod_id, quantity,
    ship_date)
    VALUES (01, 43629, 15, 'bad_data') ;
END;
```

In this example, the two INSERT statements are part of an atomic compound statement. They must either succeed or fail as one. The first INSERT statement would succeed. The second one causes a data conversion error since the value being assigned to the ship\_date column cannot be converted to a date.

The atomic compound statement fails and the effect of both INSERT statements is undone. Even if the currently executing transaction is eventually committed, neither statement in the atomic compound statement takes effect.

COMMIT and ROLLBACK and some ROLLBACK TO SAVEPOINT statements are not permitted within an atomic compound statement. See “Transactions and savepoints in procedures” on page 434.

There is a case where some, but not all, of the statements within an atomic compound statement are executed. This is when an error occurs, and is handled by an exception handler within the compound statement.

For more information, see “Using exception handlers in procedures” on page 430.

## Structure of procedures

The body of a procedure consists of a compound statement as discussed in “Using compound statements” on page 409. A compound statement consists of a BEGIN and an END, enclosing a set of SQL statements. Semicolons delimit each statement.

The SQL statements that can occur in procedures are described in “SQL statements allowed in procedures” on page 411.

Procedures can contain control statements, which are described in “Control statements” on page 408.

## SQL statements allowed in procedures

You can use almost all SQL statements within procedures, including the following:

- SELECT, UPDATE, DELETE, INSERT, and SET VARIABLE
- The CALL statement to execute other procedures
- Control statements (see “Control statements” on page 408)
- Cursor statements (see “Using cursors in procedures” on page 420)
- Exception handling statements (see “Using exception handlers in procedures” on page 430)
- The EXECUTE IMMEDIATE statement

Some SQL statements you cannot use within procedures include:

- CONNECT statement
- DISCONNECT statement

You can use COMMIT, ROLLBACK, and SAVEPOINT statements within procedures with certain restrictions (see “Transactions and savepoints in procedures” on page 434).

For details, see the Usage section for each SQL statement in Chapter 6, “SQL Statements,” in *Sybase IQ Reference Manual*.

## Declaring parameters for procedures

Procedure parameters appear as a list in the CREATE PROCEDURE statement. Parameter names must conform to the rules for other database identifiers such as column names. They must have valid data types (see Chapter 4, “SQL Data Types,” in *Sybase IQ Reference Manual*), and must be prefixed with one of the keywords IN, OUT or INOUT. These keywords have the following meanings:

- **IN** The argument is an expression that provides a value to the procedure.
- **OUT** The argument is a variable that could be given a value by the procedure.
- **INOUT** The argument is a variable that provides a value to the procedure, and could be given a new value by the procedure.

You can assign default values to procedure parameters in the CREATE PROCEDURE statement. The default value must be a constant, which may be NULL. For example, the following procedure uses the NULL default for an IN parameter to avoid executing a query that would have no meaning:

```
CREATE PROCEDURE
CustomerProducts( IN customer_id
                  INTEGER DEFAULT NULL )
RESULT ( product_id INTEGER,
         quantity_ordered INTEGER )
BEGIN
    IF customer_id IS NULL THEN
        RETURN;
    ELSE
        SELECT product.id,
               sum( sales_order_items.quantity )
        FROMproduct,
              sales_order_items,
              sales_order
        WHERE sales_order.cust_id = customer_id
        AND sales_order.id = sales_order_items.id
        AND sales_order_items.prod_id=product.id
        GROUP BY product.id;
    END IF;
END
```

The following statement causes the DEFAULT NULL to be assigned, and the procedure returns instead of executing the query.

```
CALL CustomerProducts();
```

## Passing parameters to procedures

You can take advantage of default values of stored procedure parameters with either of two forms of the CALL statement.

If the optional parameters are at the end of the argument list in the CREATE PROCEDURE statement, they may be omitted from the CALL statement. As an example, consider a procedure with three INOUT parameters:

```
CREATE PROCEDURE SampleProc( INOUT var1 INT
                            DEFAULT 1,
                            INOUT var2 int DEFAULT 2,
                            INOUT var3 int DEFAULT 3 )
    . . .
```

Assume that the calling environment has set up three variables to hold the values passed to the procedure:

```
CREATE VARIABLE V1 INT;
CREATE VARIABLE V2 INT;
CREATE VARIABLE V3 INT;
```

The procedure SampleProc may be called supplying only the first parameter as follows:

```
CALL SampleProc( V1 )
```

In this case the default values are used for *var2* and *var3*.

A more flexible method of calling procedures with optional arguments is to pass the parameters by name. The SampleProc procedure may be called as follows:

```
CALL SampleProc( var1 = V1, var3 = V3 )
```

It can also be called as follows:

```
CALL SampleProc( var3 = V3, var1 = V1 )
```

## Passing parameters to functions

User-defined functions are not invoked with the CALL statement, but are used in the same manner that built-in functions are used. For example, the following statement uses the fullname function defined in “Creating user-defined functions” on page 404 to retrieve the names of all employees:

```
SELECT fullname(emp_fname, emp_lname) AS Name
FROM employee
```

Name
Fran Whitney
Matthew Cobb
Philip Chin
Julie Jordan
Robert Breault
...

Notes

- Default parameters can be used in calling functions. However, parameters cannot be passed to functions by name.
- Parameters are passed by value, not by reference. Even if the function changes the value of the parameter, this change is not returned to the calling environment.
- Output parameters cannot be used in user-defined functions.
- User-defined functions cannot return result sets.

## Returning results from procedures

Procedures can return results that are a single row of data, or multiple rows. Results consisting of a single row of data can be passed back as arguments to the procedure. Results consisting of multiple rows of data are passed back as result sets. Procedures can also return a single value given in the RETURN statement.

For simple examples of how to return results from procedures, see “Introduction to procedures” on page 397. For more detailed information, see the following sections.

## Returning a value using the RETURN statement

The RETURN statement returns a single integer value to the calling environment, causing an immediate exit from the procedure. The RETURN statement takes the form:

RETURN *expression*

The value of the supplied expression is returned to the calling environment. To save the return value in a variable, an extension of the CALL statement is used:

```
CREATE VARIABLE returnval INTEGER ;
returnval = CALL myproc() ;
```

## Returning results as procedure parameters

Procedures can return results to the calling environment in the parameters to the procedure.

Within a procedure, parameters and variables can be assigned values using:

- The SET statement
- A SELECT statement with an INTO clause

Using the SET statement

The following somewhat artificial procedure returns a value in an OUT parameter that is assigned using a SET statement:

```
CREATE PROCEDURE greater (IN a INT,
                           IN b INT,
                           OUT c INT)
BEGIN
    IF a > b THEN
        SET c = a;
    ELSE
        SET c = b;
    END IF ;
END
```

---

**Note** The preceding example is artificial: generally a function is easier to use than a procedure when only one result is required.

---

Using single-row SELECT statements

Single-row queries retrieve at most one row from the database. This type of query is achieved by a SELECT statement with an INTO clause. The INTO clause follows the select list and precedes the FROM clause. It contains a list of variables to receive the value for each select list item. There must be the same number of variables as there are select list items.

When a SELECT statement executes, the server retrieves the results of the SELECT statement and places the results in the variables. If the query results contain more than one row, the server returns an error. For queries returning more than one row, you must use **ursors**. For information about returning more than one row from a procedure, see “Returning result sets from procedures” on page 417.

If the query results in no rows being selected, a `row not found` warning is returned.

The following procedure returns the results of a single-row SELECT statement in the procedure parameters.

To return the number of orders placed by a given customer, type the following:

```
CREATE PROCEDURE OrderCount (IN customer_ID INT,
                             OUT Orders INT)
BEGIN
    SELECT COUNT(DBA.sales_order.id)
      INTO Orders
     FROM DBA.customer
      KEY LEFT OUTER JOIN DBA.sales_order
    WHERE DBA.customer.id = customer_ID;
END
```

You can test this procedure in dbisql using the following statements, which show the number of orders placed by the customer with ID 102:

```
CREATE VARIABLE orders INT;
CALL OrderCount ( 102, orders );
SELECT orders;
```

#### Notes

- The *customer\_ID* parameter is declared as an IN parameter. This parameter holds the customer ID that is passed in to the procedure.
- The *Orders* parameter is declared as an OUT parameter. It holds the value of the *orders* variable that is returned to the calling environment.
- No DECLARE statement is required for the *Orders* variable, as it is declared in the procedure argument list.
- The SELECT statement returns a single row and places it into the variable *Orders*.

## Returning result sets from procedures

Result sets allow a procedure to return more than one row of results to the calling environment.

The following procedure returns a list of customers who have placed orders, together with the total value of the orders placed. The procedure does not list customers who have not placed orders.

```
CREATE PROCEDURE ListCustomerValue ()
RESULT ("Company" CHAR(36), "Value" NUMERIC(14,2))
BEGIN
    SELECT company_name,
           CAST( sum(sales_order_items.quantity *
                      product.unit_price)
                  AS NUMERIC(14,2)) AS value
    FROM customer
        INNER JOIN sales_order
            INNER JOIN sales_order_items
                INNER JOIN product
                    GROUP BY company_name
                    ORDER BY value DESC;
END
```

- Type the following:

```
CALL ListCustomerValue ()
```

Company	Value
Chadwicks	8076
Overland Army Navy	8064
Martins Landing	6888
Sterling & Co.	6804
Carmel Industries	6780
...	...

### Notes

- The number of variables in the RESULT list must match the number of the SELECT list items. Automatic data type conversion is carried out where possible if data types do not match.
- The RESULT clause is part of the CREATE PROCEDURE statement, and does not have a command delimiter.
- The names of the SELECT list items do not need to match those of the RESULT list.

- When testing this procedure, dbisql displays only the first result set by default. You can configure dbisql to display more than one result set by setting the Show multiple result sets option on the Commands tab of the Options dialog.
- You can modify procedure result sets, unless they are generated from a view. The user calling the procedure requires the appropriate permissions on the underlying table to modify procedure results. This is different than the usual permissions for procedure execution, where the procedure *owner* must have permissions on the table.

## Returning multiple result sets from procedures

A procedure can return more than one result set to the calling environment. If a RESULT clause is employed, the result sets must be compatible: they must have the same number of items in the SELECT lists, and the data types must all be of types that can be automatically converted to the data types listed in the RESULT list.

The method for returning multiple result sets differs for dbisql and dbisqlc.

displaying multiple result sets in dbisql

Before dbisql can return multiple result sets, you need to enable this option on the Commands tab of the Options dialog. By default, this option is disabled. If you change the setting, you must make it permanent and either start a new connection to the database or restart dbisql for the new setting to take effect.

### ❖ Enabling multiple result set functionality

- 1 Choose Tools > Options.
- 2 In the resulting Options dialog, click the Commands tab.
- 3 Select the Show Multiple Result Sets check box and click Make Permanent.
- 4 Do one of the following:  
Click File > New Window and enter connection information, or  
Restart dbisql.

displaying multiple result sets in dbisqlc

For dbisqlc, you do not enable multiple result sets in advance. Instead, you call the procedure, and after each result set is displayed in the dbisqlc data window you must enter a RESUME statement to continue, and then complete, the procedure.

**Example**

The following procedure lists the names of all employees, customers, and contacts listed in the database:

```
CREATE PROCEDURE ListPeople()
RESULT ( lname CHAR(36), fname CHAR(36) )
BEGIN
    SELECT emp_lname, emp_fname
    FROM employee;
    SELECT lname, fname
    FROM customer;
    SELECT last_name, first_name
    FROM contact;
END
```

To test this procedure in either dbisql or dbisqlc, enter the following statement:

```
CALL ListPeople ()
```

- In dbisql, if you have previously enabled multiple result sets, all three result sets appear automatically in the Results pane. Otherwise, you see only the first result set.
- In dbisqlc, to display all three result sets, enter a RESUME statement after each one is displayed. You can also do this in a batch file. For example, you could create a batch file called *listresults* with the following contents:

```
call ListPeople;
resume;
resume;
resume;
```

Then call the batch file:

```
dbisqlc -q -c "uid=DBA;pwd=SQL"
read listresults.sql > listresults.out
```

## Returning variable result sets from procedures

The RESULT clause is optional in procedures. Omitting the result clause allows you to write procedures that return different result sets, with different numbers or types of columns, depending on how they are executed.

If you do not use the variable result sets feature, you should use a RESULT clause to provide better performance, and to allow front-end tools to discern the columns and data types the procedure will produce without executing it.

For example, the following procedure returns two columns if the input variable is Y, but only one column otherwise:

```
CREATE PROCEDURE names( IN formal char(1) )
BEGIN
    IF formal = 'y' THEN
        SELECT emp_lname, emp_fname
        FROM employee
    ELSE
        SELECT emp_fname
        FROM employee
    END IF
END
```

The use of variable result sets in procedures is subject to some limitations, depending on the interface used by the client application.

- **Embedded SQL** You must DESCRIBE the procedure call after the cursor for the result set is opened, but before any rows are returned, in order to get the proper shape of result set.

For information about the DESCRIBE statement, see Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

- **ODBC** Variable result set procedures can be used by ODBC applications. The Sybase IQ ODBC driver carries out the proper description of the variable result sets.
- **Open Client applications** Open Client applications can use variable result set procedures. Sybase IQ carries out the proper description of the variable result sets.

## Using cursors in procedures

Cursors are used to retrieve rows one at a time from a query or stored procedure that has multiple rows in its result set. A **cursor** is a handle or an identifier for the query or procedure, and for a current position within the result set.

## Cursor management overview

Managing a cursor is similar to managing a file in a programming language. The following steps are used to manage cursors:

- 1 Declare a cursor for a particular SELECT statement or procedure using the DECLARE statement.
- 2 Open the cursor using the OPEN statement.
- 3 Use the FETCH statement to retrieve results one row at a time from the cursor.
- 4 The warning Row Not Found signals the end of the result set.
- 5 Close the cursor using the CLOSE statement.

By default, cursors are automatically closed at the end of a transaction (on explicit or implied COMMIT or ROLLBACK statements). Cursors that are opened using the WITH HOLD clause will be kept open for subsequent transactions until they are explicitly closed.

The sp\_iqcursorinfo stored procedure displays information about cursors currently open on the server. For more information, see “sp\_iqcursorinfo procedure” in Chapter 10, “System Procedures” of the *Sybase IQ Reference Manual*.

## Cursor positioning

A cursor can be positioned at one of three places:

- On a row
- Before the first row
- After the last row

When a cursor is opened, it is positioned before the first row. Using the FETCH command, you can move the cursor position to an absolute position from the start or the end of the query results (using FETCH ABSOLUTE, FETCH FIRST, or FETCH LAST), or to a position relative to the current cursor position (using FETCH RELATIVE, FETCH PRIOR, or FETCH NEXT). The NEXT keyword is the default qualifier for the FETCH statement.

For more information see “FETCH statement [ESQL] [SP],” in *Sybase IQ Reference Manual*.

---

**Note** Sybase IQ treats the FIRST, LAST, and ABSOLUTE options as starting from the beginning of the result set. It treats RELATIVE with a negative row count as starting from the current position.

---

There are special positioned versions of the UPDATE and DELETE statements that can be used to update or delete the row at the current position of the cursor. If the cursor is positioned before the first row or after the last row, a No current row of cursor error is returned.

## Using cursors on SELECT statements in procedures

The following procedure uses a cursor on a SELECT statement. It illustrates several features of the stored procedure language, and is based on the same query used in the ListCustomerValue procedure described in “Returning result sets from procedures”.

```
CREATE PROCEDURE TopCustomerValue
(   OUT TopCompany CHAR(36),
    OUT TopValue INT )
BEGIN
    -- 1. Declare the "error not found" exception
    DECLARE err_notfound
        EXCEPTION FOR SQLSTATE '02000';
    -- 2. Declare variables to hold
    --      each company name and its value
    DECLARE ThisName CHAR(36);
    DECLARE ThisValue INT;
    -- 3. Declare the cursor ThisCompany
    --      for the query
    DECLARE ThisCompany CURSOR FOR
        SELECT company_name,
               CAST( sum( sales_order_items.quantity *
                           product.unit_price ) AS INTEGER )
        AS value
        FROM customer
              INNER JOIN sales_order
              INNER JOIN sales_order_items
              INNER JOIN product
        GROUP BY company_name;
    -- 4. Initialize the values of TopValue
    SET TopValue = 0;
    -- 5. Open the cursor
    OPEN ThisCompany;
    -- 6. Loop over the rows of the query
    CompanyLoop:
    LOOP
        FETCH NEXT ThisCompany
        INTO ThisName, ThisValue;
```

```
IF SQLSTATE = err_notfound THEN
    LEAVE CompanyLoop;
END IF;
IF ThisValue > TopValue THEN
    SET TopCompany = ThisName;
    SET TopValue = ThisValue;
END IF;
END LOOP CompanyLoop;
-- 7. Close the cursor
CLOSE ThisCompany;
END
```

#### Notes

The `TopCustomerValue` procedure has the following notable features:

- The “error not found” exception is declared. This exception is used later in the procedure to signal when a loop over the results of a query has completed.  
For more information about exceptions, see “Errors and warnings in procedures” on page 424.
- Two local variables `ThisName` and `ThisValue` are declared to hold the results from each row of the query.
- The cursor `ThisCompany` is declared. The `SELECT` statement produces a list of company names and the total value of the orders placed by that company.
- The value of `TopValue` is set to an initial value of 0, for later use in the loop.
- The `ThisCompany` cursor is opened.
- The `LOOP` statement loops over each row of the query, placing each company name in turn into the variables `ThisName` and `ThisValue`. If `ThisValue` is greater than the current top value, `TopCompany` and `TopValue` are reset to `ThisName` and `ThisValue`.
- The cursor is closed at the end of the procedure.
- You can also write this procedure without a loop by adding an `ORDER BY` value `DESC` clause to the `SELECT` statement. Then, only the first row of the cursor needs to be fetched.

The `LOOP` construct in the `TopCompanyValue` procedure is a standard form, exiting after the last row is processed. You can rewrite this procedure in a more compact form using a `FOR` loop. The `FOR` statement combines several aspects of the above procedure into a single statement.

```
CREATE PROCEDURE TopCustomerValue2(
    OUT TopCompany CHAR(36),
```

```
        OUT TopValue INT )
BEGIN
    -- Initialize the TopValue variable
    SET TopValue = 0;
    -- Do the For Loop
    CompanyLoop:
FOR CompanyFor AS ThisCompany
    CURSOR FOR
        SELECT company_name AS ThisName ,
               CAST( sum( sales_order_items.quantity *
                           product.unit_price ) AS INTEGER )
               AS ThisValue
    FROM customer
        INNER JOIN sales_order
        INNER JOIN sales_order_items
        INNER JOIN product
    GROUP BY ThisName
DO
    IF ThisValue > TopValue THEN
        SET TopCompany = ThisName;
        SET TopValue = ThisValue;
        END IF;
    END FOR CompanyLoop;
END
```

## Errors and warnings in procedures

After an application program executes a SQL statement, it can examine a **return code** (or status code). This return code indicates whether the statement executed successfully or failed and gives the reason for the failure. You can use the same mechanism to indicate the success or failure of a CALL statement to a procedure.

Error reporting uses either the SQLCODE or SQLSTATE status descriptions. Whenever a SQL statement is executed, a value is placed in special procedure variables called SQLSTATE and SQLCODE. That value indicates whether or not there were any unusual conditions encountered while the statement was being performed. You can check the value of SQLSTATE or SQLCODE in an IF statement following a SQL statement, and take actions depending on whether the statement succeeded or failed.

For example, the SQLSTATE variable can be used to indicate if a row is successfully fetched. The TopCustomerValue procedure presented in “Using cursors on SELECT statements in procedures” on page 422 used the SQLSTATE test to detect that all rows of a SELECT statement had been processed.

For full descriptions of SQLCODE and SQLSTATE error and warning values and their meanings, see *Sybase IQ Error Messages*.

## Default error handling in procedures

This section describes how Sybase IQ handles errors that occur during a procedure execution, if you have no error handling built in to the procedure.

If you want to have different behavior, you can use exception handlers, described in “Using exception handlers in procedures” on page 430. Warnings are handled in a slightly different manner from errors: for a description, see “Default handling of warnings in procedures” on page 429.

There are two ways of handling errors without using explicit error handling:

- **Default error handling** The procedure fails and returns an error code to the calling environment.
- **ON EXCEPTION RESUME** If the ON EXCEPTION RESUME clause is included in the CREATE PROCEDURE statement, the procedure carries on executing after an error, resuming at the statement following the one causing the error.

The precise behavior for procedures that use ON EXCEPTION RESUME is dictated by the ON\_TSQL\_ERROR option setting. For more information, see “ON\_TSQL\_ERROR option [TSQL]” in Chapter 2, “Database Options,” in *Sybase IQ Reference Manual*.

### Default error handling

Generally, if a SQL statement in a procedure fails, the procedure terminates execution and control is returned to the application program with an appropriate setting for the SQLSTATE and SQLCODE values. This is true even if the error occurred in a procedure invoked directly or indirectly from the first one.

The following demonstration procedures show what happens when an application calls the procedure OuterProc, and OuterProc in turn calls the procedure InnerProc, which then encounters an error.

```
CREATE PROCEDURE OuterProc()
BEGIN
```

```
MESSAGE 'Hello from OuterProc.';
CALL InnerProc();
MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in OuterProc.'
END
CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE column_not_found
        EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from InnerProc.';
    SIGNAL column_not_found;
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in InnerProc.';
END
```

Notes

- The **DECLARE** statement in **InnerProc** declares a symbolic name for one of the predefined **SQLSTATE** values associated with error conditions already known to the server.
- The **MESSAGE** statement sends a message to the server window and the **dbisql** Messages pane.
- The **SIGNAL** statement generates an error condition from within the **InnerProc** procedure.

The following statement executes the **OuterProc** procedure:

```
CALL OuterProc();
```

The message window of the server then displays the following:

```
Hello from OuterProc.

Hello from InnerProc.
```

None of the statements following the **SIGNAL** statement in **InnerProc** execute: **InnerProc** immediately passes control back to the calling environment, which in this case is the procedure **OuterProc**. No statements following the **CALL** statement in **OuterProc** execute. The error condition is returned to the calling environment to be handled there. For example, **dbisql** handles the error by displaying a message window describing the error.

## Error handling with ON EXCEPTION RESUME

If the ON EXCEPTION RESUME clause is included in the CREATE PROCEDURE statement, the procedure checks the following statement when an error occurs. If the statement handles the error, then the procedure does not return control to the calling environment when an error occurs. Instead, it continues executing, resuming at the statement after the one causing the error.

The behavior for procedures that use ON EXCEPTION RESUME can be modified by the ON\_TSQL\_ERROR option setting. For more information, see Chapter 2, “Database Options,” of *Sybase IQ Reference Manual*.

---

**Note** When a statement has several parts or clauses, such as IF, ELSE IF, END IF, or FOR and END FOR, the “following statement” refers to the next new statement, not a statement part.

---

Error-handling statements include the following:

- IF
- SELECT @*variable* =
- CASE
- LOOP
- LEAVE
- CONTINUE
- CALL
- EXECUTE
- SIGNAL
- RESIGNAL
- DECLARE
- SET *variable*

The following example illustrates how this works.

Drop the procedures

Remember to drop both the OuterProc and InnerProc procedures by entering the following commands in the command window before continuing with the tutorial:

```
DROP PROCEDURE OUTERPROC;  
DROP PROCEDURE INNERPROC
```

The following demonstration procedures show what happens when an application calls the procedure OuterProc; and OuterProc in turn calls the procedure InnerProc, which then encounters an error. These demonstration procedures are based on those used earlier in this section:

```
CREATE PROCEDURE OuterProc()
ON EXCEPTION RESUME
BEGIN
    DECLARE res CHAR(5);
    MESSAGE 'Hello from OuterProc.';
    CALL InnerProc();
    SELECT @res=SQLSTATE;
    IF @res='52003' THEN
        MESSAGE 'SQLSTATE set to ',
               res, ' in OuterProc.';
    END IF
END;

CREATE PROCEDURE InnerProc()
ON EXCEPTION RESUME
BEGIN
    DECLARE column_not_found
        EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from InnerProc.';
    SIGNAL column_not_found;
    MESSAGE 'SQLSTATE set to ',
           SQLSTATE, ' in InnerProc.';
END
```

The following statement executes the OuterProc procedure:

```
CALL OuterProc();
```

The message window of the server or the dbisql Messages pane then displays the following:

```
Hello from OuterProc.
Hello from InnerProc.
SQLSTATE set to 52003 in OuterProc.
```

The execution path is as follows:

- 1 OuterProc executes and calls InnerProc.
- 2 In InnerProc, the SIGNAL statement signals an error.
- 3 The MESSAGE statement is not an error-handling statement, so control is passed back to OuterProc and the message is not displayed.

- 4 In OuterProc, the statement following the error assigns the SQLSTATE value to the variable named res. This is an error-handling statement, and so execution continues and the OuterProc message is displayed.

## Default handling of warnings in procedures

Errors and warnings are handled differently. While the default action for errors is to set a value for the SQLSTATE and SQLCODE variables, and return control to the calling environment in the event of an error, the default action for warnings is to set the SQLSTATE and SQLCODE values and continue execution of the procedure.

### Drop the procedures

Remember to drop both the InnerProc and OuterProc procedures by entering the following commands in the command window before continuing with the tutorial:

```
DROP PROCEDURE OUTERPROC;
DROP PROCEDURE INNERPROC
```

The following demonstration procedures illustrate default handling of warnings. These demonstration procedures are based on those used in “Default error handling in procedures” on page 425. In this case, the SIGNAL statement generates a row not found condition, which is a warning rather than an error.

```
CREATE PROCEDURE OuterProc()
BEGIN
    MESSAGE 'Hello from OuterProc.';
    CALL InnerProc();
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in OuterProc.';
END
CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE row_not_found
        EXCEPTION FOR SQLSTATE '02000';
    MESSAGE 'Hello from InnerProc.';
    SIGNAL row_not_found;
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE, ' in InnerProc.';
END
```

The following statement executes the OuterProc procedure:

```
CALL OuterProc();
```

The message window of the server or the dbisql Messages pane then displays the following:

```
Hello from OuterProc.
```

```
Hello from InnerProc.
```

```
SQLSTATE set to 02000 in InnerProc.
```

```
SQLSTATE set to 00000 in OuterProc.
```

The procedures both continued executing after the warning was generated, with SQLSTATE set by the warning (02000).

Execution of the second MESSAGE statement in InnerProc resets the warning. Successful execution of any SQL statement resets SQLSTATE to 00000 and SQLCODE to 0. If a procedure needs to save the error status, it must do an assignment of the value immediately after execution of the statement which caused the error warning.

## Using exception handlers in procedures

It is often desirable to intercept certain types of errors and handle them within a procedure, rather than pass the error back to the calling environment. This is done through the use of an **exception handler**.

You define an exception handler with the EXCEPTION part of a compound statement (see “Using compound statements” on page 409). Whenever an error occurs in the compound statement, the exception handler executes. Unlike errors, warnings do not cause exception handling code to be executed. Exception handling code also executes if an error appears in a nested compound statement or in a procedure invoked anywhere within the compound statement.

### Drop the procedures

Remember to drop both the InnerProc and OuterProc procedures by entering the following commands in the command window before continuing with the tutorial:

```
DROP PROCEDURE OUTERPROC;  
DROP PROCEDURE INNERPROC
```

The demonstration procedures used to illustrate exception handling are based on those used in “Default error handling in procedures” on page 425. In this case, additional code handles the column not found error in the InnerProc procedure.

```
CREATE PROCEDURE OuterProc()
BEGIN
    MESSAGE 'Hello from OuterProc.';
    CALL InnerProc();
    MESSAGE 'SQLSTATE set to ',
        SQLSTATE,' in OuterProc.'
END
CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE column_not_found
        EXCEPTION FOR SQLSTATE '52003';
    MESSAGE 'Hello from InnerProc.';
    SIGNAL column_not_found;
    MESSAGE 'Line following SIGNAL.';
    EXCEPTION
        WHEN column_not_found THEN
            MESSAGE 'Column not found handling.';
        WHEN OTHERS THEN
            RESIGNAL ;
    END

```

The EXCEPTION statement declares the exception handler itself. The lines following the EXCEPTION statement do not execute unless an error occurs. Each WHEN clause specifies an exception name (declared with a DECLARE statement) and the statement or statements to be executed in the event of that exception. The WHEN OTHERS THEN clause specifies the statement(s) to be executed when the exception that occurred does not appear in the preceding WHEN clauses.

In this example, the statement RESIGNAL passes the exception on to a higher-level exception handler. RESIGNAL is the default action if WHEN OTHERS THEN is not specified in an exception handler.

The following statement executes the OuterProc procedure:

```
CALL OuterProc();
```

The message window of the server or the dbisql Messages pane then displays the following:

```
Hello from OuterProc.
Hello from InnerProc.
Column not found handling.
SQLSTATE set to 00000 in OuterProc.
```

#### Notes

- The EXCEPTION statements execute, rather than the lines following the SIGNAL statement in InnerProc.

- As the error encountered was a `column not found` error, the `MESSAGE` statement included to handle the error executes, and `SQLSTATE` resets to zero (indicating no errors).
- After the exception handling code is executed, control is passed back to `OuterProc`, which proceeds as if no error was encountered.
- You should not use `ON EXCEPTION RESUME` together with explicit exception handling. The exception handler code is not executed if `ON EXCEPTION RESUME` is included.
- You should use explicit exception handling code after each statement that may potentially generate an exception whenever you use `ON EXCEPTION RESUME`. You gain flexibility in handling errors, but the cost is more code and a higher risk of bugs in your code.
- If the error handling code for the `column not found` exception is simply a `RESIGNAL` statement, control is passed back to the `OuterProc` procedure with `SQLSTATE` still set at the value 52003. This is just as if there were no error handling code in `InnerProc`. Since there is no error handling code in `OuterProc`, the procedure fails.

**Exception handling  
and atomic compound  
statements**

When an exception is handled inside a compound statement, the compound statement completes without an active exception and the changes before the exception are not reversed. This is true even for atomic compound statements. If an error occurs within an atomic compound statement and is explicitly handled, some but not all of the statements in the atomic compound statement are executed.

## Nested compound statements and exception handlers

The code following a statement that causes an error is not executed unless an `ON EXCEPTION RESUME` clause is included in a procedure definition.

You can use nested compound statements to give you more control over which statements execute following an error and which do not.

**Drop the procedures**

Remember to drop both the `InnerProc` and `OuterProc` procedures by entering the following commands in the command window before continuing with the tutorial:

```
DROP PROCEDURE OUTERPROC;  
DROP PROCEDURE INNERPROC
```

The following demonstration procedure illustrates how nested compound statements can be used to control flow. The procedure is based on that used as an example in “Default error handling in procedures” on page 425

```
CREATE PROCEDURE InnerProc()
BEGIN
    DECLARE column_not_found
        EXCEPTION FOR SQLSTATE VALUE '52003';
        MESSAGE 'Hello from InnerProc';
    BEGIN
        SIGNAL column_not_found;
        MESSAGE 'Line following SIGNAL'
    EXCEPTION
        WHEN column_not_found THEN
            MESSAGE 'Column not found handling';
        WHEN OTHERS THEN
            RESIGNAL;
    END
    MESSAGE 'Outer compound statement';
END
```

The following statement executes the InnerProc procedure:

```
CALL InnerProc();
```

The message window of the server or the dbisql Messages pane then displays the following:

```
Hello from InnerProc
Column not found handling
Outer compound statement
```

When the SIGNAL statement that causes the error is encountered, control passes to the exception handler for the compound statement, and the Column not found handling message is printed. Control then passes back to the outer compound statement and the Outer compound statement message is printed.

If an error other than column not found is encountered in the inner compound statement, the exception handler executes the RESIGNAL statement. The RESIGNAL statement passes control directly back to the calling environment, and the remainder of the outer compound statement is not executed.

## Using the EXECUTE IMMEDIATE statement in procedures

The EXECUTE IMMEDIATE statement allows statements to be built up inside procedures using a combination of literal strings (in quotes) and variables.

For example, the following procedure includes an EXECUTE IMMEDIATE statement that creates a table.

```
CREATE PROCEDURE CreateTableProc(
    IN tablename char(30) )
BEGIN
    EXECUTE IMMEDIATE 'CREATE TABLE ' || tablename ||
    (column1 INT PRIMARY KEY)'
END
```

In ATOMIC compound statements, you cannot use an EXECUTE IMMEDIATE statement that causes a COMMIT, as COMMITs are not allowed in that context.

## Transactions and savepoints in procedures

SQL statements in a procedure are part of the current transaction (see Chapter 10, “Transactions and Versioning”). You can call several procedures within one transaction or have several transactions in one procedure.

COMMIT and ROLLBACK are not allowed within any atomic statement (see “Atomic compound statements” on page 410).

Savepoints (see “Savepoints within transactions” on page 493) can be used within a procedure, but a ROLLBACK TO SAVEPOINT statement can never refer to a savepoint before the atomic operation started. Also, all savepoints within an atomic operation are released when the atomic operation completes.

## Tips for writing procedures

This section provides some pointers for developing procedures.

## Check if you need to change the command delimiter

You do not need to change the command delimiter in dbisql or Sybase Central when you are writing procedures. However, if you create and test procedures from some other browsing tool, you may need to change the command delimiter from the semicolon to another character.

Each statement within the procedure ends with a semicolon. For some browsing applications to parse the CREATE PROCEDURE statement itself, you need the command delimiter to be something other than a semicolon.

If you are using an application that requires changing the command delimiter, a good choice is to use two semicolons as the command delimiter (;;) or a question mark (?) if the system does not permit a multicharacter delimiter.

## Remember to delimit statements within your procedure

You should terminate each statement within the procedure with a semicolon. Although you can leave off semicolons for the last statement in a statement list, it is good practice to use semicolons after each statement.

The CREATE PROCEDURE statement itself contains both the RESULT specification and the compound statement that forms its body. No semicolon is needed after the BEGIN or END keywords, or after the RESULT clause.

## Use fully-qualified names for tables in procedures

If a procedure has references to tables in it, you should always preface the table name with the name of the owner (creator) of the table.

When a procedure refers to a table, it uses the group memberships of the procedure creator to locate tables with no explicit owner name specified. For example, if a procedure created by user\_1 references Table\_B and does not specify the owner of Table\_B, then either Table\_B must have been created by user\_1 or user\_1 must be a member of a group (directly or indirectly) that is the owner of Table\_B. If neither condition is met, a `table not found` message results when the procedure is called.

You can minimize the inconvenience of long fully qualified names by using a correlation name to provide a convenient name to use for the table within a statement. Correlation names are described in “FROM clause” in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

## Specifying dates and times in procedures

When dates and times are sent to the database from procedures, they are sent as strings. The date part of the string is interpreted according to the current setting of the DATE\_ORDER database option. As different connections may set this option to different values, some strings may be converted incorrectly to dates, or the database may not be able to convert the string to a date.

You should use the unambiguous date format `yyyy-mm-dd` or `yyyy/mm/dd` when using data strings within procedures. The server interprets these strings unambiguously as dates, regardless of the DATE\_ORDER database option setting.

For more information on dates and times, see “Date and time data types” in Chapter 4, “SQL Data Types,” in the *Sybase IQ Reference Manual*.

## Verifying procedure input arguments are passed correctly

One way to verify input arguments is to display the value of the parameter in the dbisql Messages pane (or the message window of the server) using the MESSAGE statement. For example, the following procedure simply displays the value of the input parameter `var`:

```
CREATE PROCEDURE message_test (IN var char(40))
BEGIN
    MESSAGE var TO CLIENT;
END
```

You can also use the stored procedure debugger.

See “MESSAGE statement” in Chapter 6, “SQL Statements,” of the *Sybase IQ Reference Manual* for information on determining the destination of the MESSAGE statement output.

## Hiding the contents of procedures, functions, and views

In some cases, you may want to distribute an application and a database without disclosing the logic contained within procedures, functions, triggers and views. As an added security measure, you can obscure the contents of these objects using the SET HIDDEN clause of the ALTER PROCEDURE, ALTER FUNCTION, and ALTER VIEW statements.

The SET HIDDEN clause scrambles the contents of the associated objects and makes them unreadable, while still allowing the objects to be used. You can also unload and reload the objects into another database.

The modification is irreversible, and for databases created using Sybase IQ 12.6 or higher, deletes the original text of the object. Preserving the original source for the object outside the database is required.

Debugging using the stored procedure debugger will not show the procedure definition, nor will procedure profiling display the source.

Running one of the above statements on an object that is already hidden has no effect.

To hide the text for all objects of a particular type, you can use a loop similar to the following:

```
begin
    for hide_lp as hide_cr cursor for
        select proc_name,user_name
        from SYS.SYSPROCEDURE p, SYS.SYSUSERPERM u
        where p.creator = u.user_id
        and p.creator not in (0,1,3)
    do
        message 'altering ' || proc_name;
        execute immediate 'alter procedure "' ||
            user_name || '.' || proc_name
            || ' set hidden'
    end for
end
```

For more information, see “ALTER FUNCTION statement,” in the *Adaptive Server Anywhere SQL Reference Manual*, “ALTER PROCEDURE statement,” and “ALTER VIEW statement” in the *Sybase IQ Reference Manual*.

## Statements allowed in batches

All SQL statements are acceptable in batches (including data definition statements such as CREATE TABLE, ALTER TABLE, and so on), with the exception of the following:

- CONNECT or DISCONNECT statement
- ALTER PROCEDURE or ALTER FUNCTION statement

You also cannot use host variables in batches.

The CREATE PROCEDURE statement is allowed, but must be the final statement of the batch. Therefore a batch can contain only a single CREATE PROCEDURE statement.

## Using SELECT statements in batches

You can include one or more SELECT statements in a batch.

The following is a valid batch:

```
IF EXISTS(SELECT *
          FROM systable
          WHERE table_name='employee' )
THEN
    SELECTTemp_lname AS LastName,
           emp_fname AS FirstName
    FROM employee;
    SELECT lname, fname
    FROM customer;
    SELECT last_name, first_name
    FROM contact;
END IF
```

The alias for the result set is required only in the first SELECT statement, as the server uses the first SELECT statement in the batch to describe the result set.

A RESUME statement is required following each query to retrieve the next result set.

## Using IQ UTILITIES to create your own stored procedures

The system stored procedures provided in Sybase IQ are implemented in SQL, using the methods described in the rest of this chapter. You may wish to create your own variants of some of these procedures. Among the ways you might do this are:

- Create a procedure that calls a system stored procedure. The `sp_iq_process_login` procedure in *Sybase IQ Reference Manual* is an example of this method.

- Create a procedure that is independent of the system stored procedures but performs a similar function.
- Create a procedure that uses the same structure as the system stored procedures but provides additional functionality. For example, you might want to display procedure results in graphical form in a front-end tool or browser rather than as text.

If you choose the second or third option, you need to understand the IQ UTILITIES statement and the strict requirements for using it.

## How IQ uses the IQ UTILITIES command

IQ UTILITIES is the underlying statement that executes whenever you run most IQ system procedures. In most cases, users are unaware that IQ UTILITIES is executing. The only time IQ UTILITIES is issued directly by users is to run the IQ buffer cache monitor.

IQ UTILITIES provides a systematic way to collect and report on information maintained in the IQ system tables. There is no general user interface; you can only use IQ UTILITIES in the ways that existing system procedures do.

System procedures declare local temporary tables in which they store information. They execute IQ UTILITIES to get the information from the system tables and store this information in the local temporary table. The system procedures may simply report the information from the local temporary table or perform additional processing.

In some system procedures, the IQ UTILITIES statement includes a predefined number as one of its arguments. This number performs a specific function, for example, deriving a value from information in the system tables. See Table 8-1 on page 441 for a list of the numbers used as IQ UTILITIES arguments.

## Requirements for using IQ UTILITIES

Be sure to read this entire chapter. The requirements discussed throughout this chapter also apply when writing stored procedures using IQ UTILITIES. You must obey several crucial additional requirements.

If you use IQ UTILITIES in your procedure, you must use this statement in exactly the same way that existing procedures use it. In particular, you must:

- Declare a local temporary table in which you store results from the procedure. This table must have exactly the same schema as in the system stored procedures, including column names, column width, column order, data types, precision, and so on.
- Issue an EXECUTE IMMEDIATE command to execute IQ UTILITIES and store its results in the temporary table.
- Where the IQ UTILITIES statement includes a number, you must use exactly the same number as in the system stored procedures, for exactly the same purpose. You cannot create your own numbers or change the way in which existing numbers are used.

In other words, you must use the local temporary table and IQ UTILITIES statement in exactly the same way as system stored procedures.

- Do not eliminate columns or add extra columns.
- Do not alter the contents of the table used in the system procedures. Users who call your procedure may also call other procedures that use the same table.

---

**Warning!** You must obey the rules listed here. Violating these rules can cause serious problems for your IQ server or database.

---

Location of system procedures

IQ system procedures are in the file *iqprocs.sql* in the *scripts* directory of your IQ installation directory.

IQ UTILITIES syntax

The syntax for IQ UTILITIES is:

**IQ UTILITIES MAIN INTO** *local-temp-table-name arguments*

For examples of how this command is used, refer to the *iqprocs.sql* file.

The IQ UTILITIES command is not documented in the *Sybase IQ Reference Manual* except to run the IQ monitor, because of the strict requirements for its use and the risk to system operations if it is used incorrectly.

Consistency across versions

The numbers in IQ system procedures are fixed. They will not change from release to release, although new numbers may be added in future releases.

Naming your procedures

Give your procedures a different name from the system procedures.

## Choosing procedures to call

You can safely use IQ UTILITIES to create your own versions of documented system procedures that report on information in the database. For example, sp\_iqspaceused displays information about used and available space available in the IQ Main and IQ Temporary Stores. Check the owner of the procedure you create from a system stored procedure to be sure your version of the procedure has the correct owner.

*Do not* create your own versions of system procedures that control IQ or multiplex operations. Modifying procedures that control IQ operations or multiplex administration can lead to serious problems.

## Numbers used by IQ UTILITIES

The following table lists the numbers used as arguments in the IQ UTILITIES command and the system procedure where each number is used. For information on the function of these procedures, see Chapter 10, “System Procedures” in the *Sybase IQ Reference Manual*.

**Table 8-1: IQ UTILITIES values used in system procedures**

Number	Procedure	Comments
10000	sp_iqtransaction	
20000	sp_iqconnection and sp_iqmpxcountdbremote	
30000	sp_iqspaceused	
40000	sp_iqspaceinfo	
50000	sp_iqlocks	
60000	sp_iqmpxversionfetch	Do Not Use
70000	sp_iqmpxdumpitvlog	
80000	sp_iqcontext	
100000	sp_iqindexfragmentation	
110000	sp_iqrowdensity	

## Testing your procedures

Always test your procedures in a development environment first. Testing procedures before you run them in a production environment helps maintain the stability of your IQ server and database.



# Ensuring Data Integrity

## About this chapter

This chapter describes facilities for ensuring that the data in your database is valid and reliable. These facilities include constraints on tables and columns, and choosing appropriate data types.

The SQL statements in this chapter use the CREATE TABLE statement and ALTER TABLE statement, basic forms of which were introduced in Chapter 5, “Working with Database Objects.”

## Contents

Topic	Page
Data integrity overview	443
Using column defaults	447
USING table and column constraints	455
Declaring entity and referential integrity	460
Integrity rules in the system tables	469

## 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 Chapter 8, “Using Procedures and Batches”. For information on JDBC see Appendix B, “Data Access Using JDBC.”

## 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.

- |                                   |   |
|-----------------------------------|---|
| Inconsistent schema               | <ul style="list-style-type: none"><li>An operator enters orders to an orders table for a customer_id that does not exist in the customers table.</li></ul>  |
| Incorrectly formatted information | <ul style="list-style-type: none"><li>An operator enters text where numeric data is required.</li><li>An operator enters numeric data that is too wide for the column.</li></ul>  |
| Duplicated data                   | <ul style="list-style-type: none"><li>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.</li></ul> |

## Ensuring 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.

See “Creating tables” on page 239 for more information on how constraints affect optimization. For more on join indexes and foreign keys, see “Using join indexes” on page 287.

#### 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.

## Changing 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 item in the 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.

These and other table and column constraints are discussed in “USING table and column constraints” on page 455. 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:

- **Entity integrity** 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.
- **Referential integrity** 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.

For more information about referential integrity, see “Declaring entity and referential integrity” on page 460.

## SQL statements for implementing integrity constraints

The following SQL statements are used to implement integrity constraints:

- **CREATE TABLE statement** This statement implements integrity constraints as the database is being created.
- **ALTER TABLE statement** This statement adds integrity constraints, or deletes constraints, from an existing database.

For full descriptions of the syntax of these statements, see Chapter 6, “SQL Statements” in *Sybase IQ Reference Manual*.

## Using column defaults

Column defaults automatically assign a specified value to particular 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.

### 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
- 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

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 about the DEFAULT value of a column.

Examples

The following statement creates 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)
```

The following statement adds 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
```

The following statement defines a domain named dom1 with a data type of INTEGER and a default value of 45:

```
CREATE DOMAIN dom1 INTEGER DEFAULT 45
```

Each of the other default values is specified in a similar manner. For more information, see ALTER TABLE statement, CREATE TABLE statement, and CREATE DOMAIN statement in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

## Modifying and deleting column defaults

You can change or remove column defaults using 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
```

You can remove column defaults by modifying them to be NULL. The following statement removes the default from the order\_date column:

```
ALTER TABLE sales_order  
MODIFY order_date DEFAULT NULL
```

## Working with column default values

Sybase IQ supports loading and inserting column default values using 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. See Table 7-8 on page 370 for information on supported implicit data type conversions.
- 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.
- If the LOAD TABLE DEFAULTS option is ON (the default) and the column has a default value, that value is used. If the DEFAULTS option is OFF, any column not present in the column list is assigned NULL. The setting for the LOAD TABLE DEFAULTS parameter applies to all column DEFAULT values, including AUTOINCREMENT.
- 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.

See the individual sections for specific default value types later in this section for more information on defining and inserting column default values. Also see “Special values” in Chapter 3, “SQL Language Elements” of the *Sybase IQ Reference Manual* for more information on the special values that can be used in default column value expressions.

## Working with 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.

### ❖ Displaying the property sheet for a column

- 1 Connect to the database.
- 2 Open the Tables folder for that database.
- 3 Double-click the table holding the column you want to change.
- 4 In the right pane, click the Columns tab.
- 5 Select the desired column.
- 6 Choose File > Properties.

The property sheet of the column appears.

---

**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.

TIMESTAMP default

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.

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.

For more information about timestamps, times, and dates, see Chapter 4, “SQL Data Types” in the *Sybase IQ Reference Manual*.

## 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 Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*). 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. For more information, see Chapter 3, “SQL Language Elements” in the *Sybase IQ Reference Manual*.

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 `NONE` default

For columns that allow `NONE` values, specifying a `NONE` 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 `NONE` value.

You can use `NONE` defaults when information for some columns is optional or not always available, and when it is not required for the data in the database be correct.

For more information about the `NONE` value, see “`NONE` value” in Chapter 3, “SQL Language Elements” of the *Sybase IQ Reference Manual*.

## 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() ) )
```

## USING table and column constraints

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 tables can interfere with other users of the database. 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. For large tables, ALTER TABLE can be a time-consuming operation, and no other requests referencing the table being altered are allowed while the statement is being processed.

---

This section describes how to use constraints to help ensure that the data entered in the table is correct, and to provide information to Sybase IQ that boosts performance.

## Using 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)
```

## Using IQ UNIQUE constraint 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.

## Using 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.

#### 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.

## Defining 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 positint INT
CHECK ( @col > 0 )
```

Any variable name prefixed with @ could be used instead of @col. Any column defined using the positint 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 “Domains” in Chapter 4, “SQL Data Types,” of the *Sybase IQ Reference Manual*.

## Working with 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.

- ❖ **Displaying the property sheet for a column**

- 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 Double-click the Columns folder to open it, and double-click the column to display its property sheet.

For more information, see the Sybase Central online Help.

## Using CHECK conditions on tables

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.

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)
```

## Modifying and deleting CHECK conditions

There are several ways to alter the existing set of CHECK conditions on a table.

- You can add a new CHECK condition to the table or to an individual column, as described above.
- You can delete a CHECK condition on a column by setting 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
```

- 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' ) )
```

There are two ways of modifying a CHECK condition defined on the table, as opposed to a CHECK condition defined on a column.

- You can add a new CHECK condition using ALTER TABLE with an ADD table-constraint clause.
- You can delete all existing CHECK conditions, including column CHECK conditions, using ALTER TABLE DELETE CHECK, and then add in new CHECK conditions.

All CHECK conditions on a table, including CHECK conditions on all its columns and CHECK conditions inherited from user-defined data types, are removed using the ALTER TABLE statement with the DELETE CHECK clause, as follows:

```
ALTER TABLE table_name  
DELETE CHECK
```

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 will produce a column not found error message.

## Declaring 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.

### Declaring 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. For details, see ALTER TABLE statement in *Sybase IQ Reference Manual*.

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.

For information on creating primary keys, see “Creating primary and foreign keys” on page 247. For the detailed syntax of the CREATE TABLE statement, see CREATE TABLE statement. For information about changing table structure, see ALTER TABLE statement. Both statements are described in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

## Enforcing 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 employee table in the sample 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 the *Introduction to Sybase IQ* for a diagram of the sample database structure.

**Example 2** The sales\_order\_items table in the sample 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.

## **Declaring 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.

## **Defining foreign keys**

Use the CREATE TABLE statement or ALTER TABLE statement to create foreign keys, as you do primary keys.

---

**Note** You cannot create foreign key constraints on local temporary tables. Global temporary tables must be created with ON COMMIT PRESERVE ROWS.

---

For information on creating foreign keys, see “Creating primary and foreign keys” on page 247.

### **Example**

The sample 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.

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. For more information, see “Disabling referential integrity checking” on page 469.

❖ **Enforcing referential integrity with existing unenforced foreign keys**

- 1 Identify the candidate key to foreign key relationship.

In the preceding schema, there are two such relationships:

- Foreign key(EMPLOYEE.DeptNo to Candidate key(DEPT.DeptNo)
- Foreign key(DEPT.Mgr) to Candidate key (EMPLOYEE.EMPNo)

- 2 Add a primary key or unique constraint on the candidate key via the ALTER TABLE statement if none exist. (In the preceding example, the primary key already exists.) All candidate key values must be unique and non-null.
- 3 Drop 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);
```

**Example 3**

If you are creating a new table, you enforce referential integrity as follows:

❖ **Enforcing referential integrity in a new table**

- 1 Create the primary table, for example:

```
CREATE TABLE DEPT(DeptNo int primary key,
DeptName varchar(20),
Mgr int );
```

- 2 Create the foreign table. For example, in this statement, the default role name for the specified foreign key is DEPT:

```
CREATE TABLE EMPLOYEE (EmpNo int primary key,
DeptNo int references DEPT (DeptNo)
on delete restrict,
LastName varchar(20),
FirstName varchar(20),
Salary int);
```

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));
```

- 3 Add the foreign key constraint. For example:

```
ALTER TABLE DEPT ADD FOREIGN KEY MGR_EMPNO (Mgr)
REFERENCES EMPLOYEE (EmpNo);
```

**Example 4**

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, as shown in Chapter 10, “System Procedures,” in the *Sybase IQ Reference Manual*.

- 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 5

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 6

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 7

Suppose 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.

## Losing referential integrity

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.

## Controlling 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 9-1: Concurrent operations that return an ASIQ 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.

Table 9-2 shows which operations on both foreign table and primary table should be allowed and which return an error. Table 9-2 applies only to *enforced* foreign keys and candidate key.

**Table 9-2: 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

	<b>LOAD or INSERT ftab1</b>	<b>DELETE/ TRUNCATE TABLE ftab1</b>	<b>UPDATE ftab1 (fk1,fk2)</b>	<b>Populate new index non-FK ftab1 (fk1,fk2)</b>	<b>ADD FK ftab1 (fk1 fk2)</b>	<b>DROP FK ftab1 (fk2, fk2)</b>
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.

---

## Disabling 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. For more information, see “DISABLE\_RI\_CHECK option” in Chapter 2, “Database Options” of the *Sybase IQ Reference Manual*.

## 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 Chapter 9, “System Tables,” in the *Sybase IQ Reference Manual*. You can use Sybase Central or dbisql to browse these tables and views.



# Transactions and Versioning

## About this chapter

This chapter describes the Sybase IQ approach to transaction processing, called snapshot versioning, and its implications for performance and other aspects of database administration.

## Contents

Topic	Page
Overview of transactions and versioning	471
Versioning prevents inconsistencies	485
How locking works	485
Managing locks	489
Isolation levels	490
Checkpoints, savepoints, and transaction rollback	492
Performance implications	498
Cursors in transactions	500

## Overview of transactions and versioning

Sybase IQ uses *transaction processing* to allow many users to read from the database while it is being updated. Transaction processing ensures that logically related commands are executed as a unit. 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.

## Introduction to transactions

*Transactions* are simply 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.

### Transactions are 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 all 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. In DBISQL the AUTO\_COMMIT option can be used to control commits and rollbacks automatically.

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 Adaptive Server 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 the *Adaptive Server Anywhere User's Guide* for information on transactions and locking.

---

## Using 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.

## Starting transactions

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.

## Completing 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, depending on whether the DBISQL option COMMIT\_ON\_EXIT is set.
- 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.

## Options in DBISQL

DBISQL provides two options that let you control when and how transactions end:

- If you set the option AUTO\_COMMIT to ON, DBISQL automatically commits your results following every successful statement, and automatically performs a ROLLBACK after each failed statement.
- The setting of the option COMMIT\_ON\_EXIT controls what happens to uncommitted changes when you exit DBISQL. If this option is set to ON (the default), DBISQL does a COMMIT; otherwise it undoes your uncommitted changes with a ROLLBACK statement.

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. For further information, see Appendix A, “Compatibility with Other Sybase Databases,” in the *Sybase IQ Reference Manual*.

## **Committing a transaction writes data to 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. Reading from and writing to the cache reduces the number of times Sybase IQ must access the disk. It 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.

## **Subdividing transactions**

You can identify important states within a transaction and return to them selectively or cause other actions to occur by using savepoints. Savepoints are discussed further in “Savepoints within transactions” on page 493.

## **Displaying information about transactions**

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. For more information, see “`sp_iqtransaction` procedure” on page 838 in the *Sybase IQ Reference Manual*.

## **Introduction to concurrency**

Sybase IQ can execute more than one transaction at the same time. The term *concurrency* refers to this ability. Special mechanisms within the database server allow IQ transactions to execute concurrently without interfering with each other.

## **How concurrency works in IQ**

While executing the SQL statements that comprise one transaction, the database server can execute some or all of the statements in other transactions. Transactions processed at the same time are said to be concurrent.

Sybase IQ's approach to concurrency is designed especially 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, described in “Introduction to 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.

## **Concurrency and IQ Multiplex**

IQ Multiplex extends Sybase IQ to allow concurrent processing of read transactions on multiple Sybase IQ servers. IQ Multiplex extends snapshot versioning to maintain the consistency of the database across multiple servers.

## **Concurrency for backups**

Backup is a DML operation. Backup backs up as of the start of the backup command (the checkpoint). Backups may be performed concurrently with read and write operations. Restore operations, however, require exclusive access, because they write to the database. See Chapter 14, “Data Backup, Recovery, and Archiving” for more information on concurrency issues for backup and restore operations.

## **Why concurrency benefits you**

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.

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.

## Multiplex table version logging

By allowing only a single write server, Sybase IQ multiplex eliminates the overhead and scalability limits of a distributed lock manager. As it completes transactions that modify data in the IQ Store, the write server must communicate information about new table versions to the query servers. A multiplex accomplishes this using **table version logging** while the servers continue to run.

Query servers continue to run while the write server issues DDL operations and the DDL operations are logged in the table version log (TLV log). The write server preserves older table versions for as long as needed. In single node mode, no version logging takes place. Query servers cannot run and must be synchronized to restart.

## Communications between servers

Information about new schema and table versions propagates through the Main IQ Store. To help recover space from old versions of database objects, IQ uses a data-replication tool called SQL Remote. It lets the write server recover space when it is no longer needed by any server in the multiplex.

Before it can reclaim space used by old versions of database objects, the write server requires information about version use in the query servers. IQ uses the data store replicated to SQL Remote to gather this information.

SQL Remote exchanges data between servers using **messages**. Each message carries its destination address and other control information, so that no direct connection is needed between servers exchanging information.

## The Message Agent

The SQL Remote **Message Agent** is a client application that sends and receives messages from database to database. The installer automatically installs the Message Agent for each write and query server that you create.

The Message Agent is a program called *dbremote.exe* on Windows systems, and *dbremote* on UNIX. Sybase Central starts the Message Agent on the same node as the write server when you start a server in an Sybase IQ multiplex database. For more details about the Message Agent, see “Multiplex options” on page 5 in *Sybase IQ Utility Guide*. One dbremote process for each server runs on the write server’s node. Each dbremote process connects to its designated server to update its catalog. To start or stop dbremote processes, see “Managing multiplex servers” on page 207.

## Introduction to versioning

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.

## Table-level versioning

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.

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.

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.

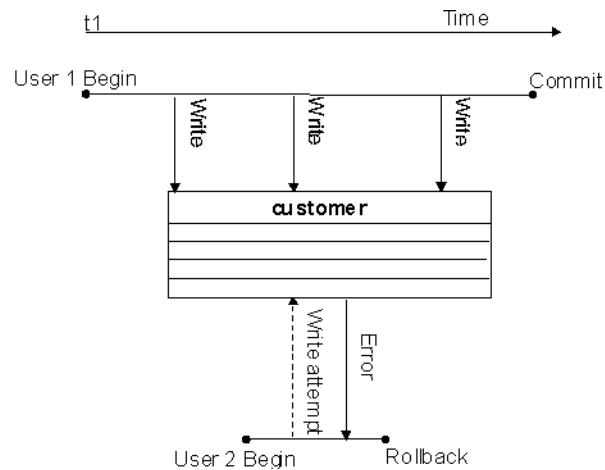
Sybase IQ's table level versioning is extended for its multiplex databases. When a transaction creating a new version of a table commits on the write server, the control information describing that new version is available to all the query servers instantly. New transactions beginning on the query server automatically see these new versions of the tables, just as new transactions on the write server 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.

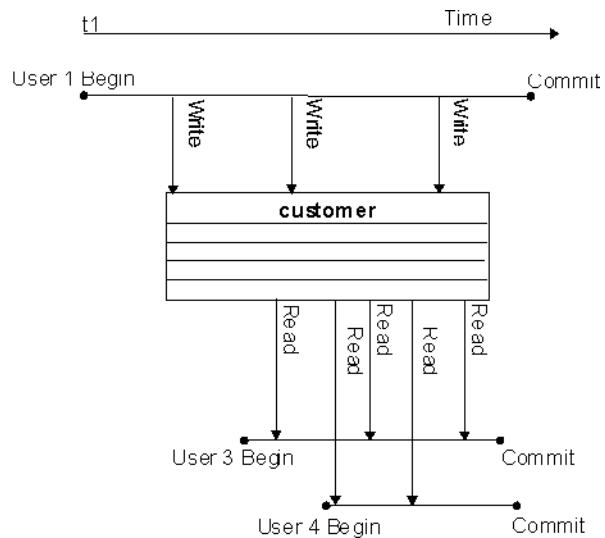
Imagine a situation such as the one shown in Figure 10-1. First, User 1 begins a transaction and starts to insert data into the customer table. As long as User 1's transaction remains open, no other user can write to the customer table. Any transaction that attempts to write to the customer table receives an error until User 1's transaction commits.

In Figure 10-1, 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 10-1: Only one writer at a time**

Meanwhile, other users can read from the customer table at any time. In this way queries can proceed while the database administrator inserts and deletes table data. In Figure 10-2, User 3 and User 4 are able to query the customer table while User 1's write transaction remains open.

**Figure 10-2: One writer, multiple readers**

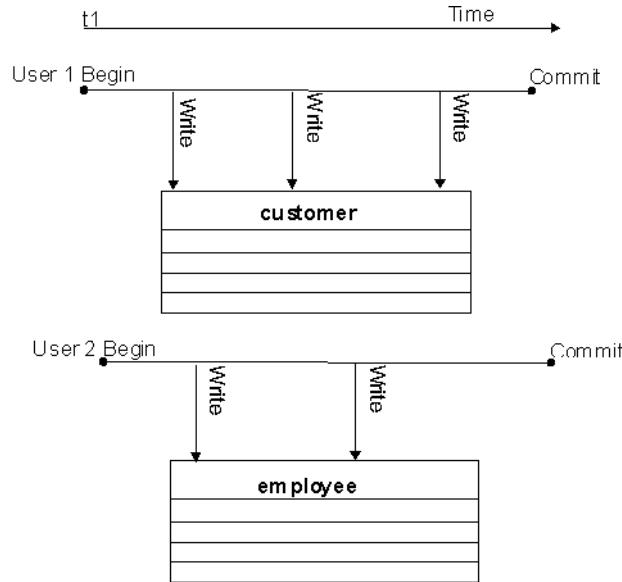


## Multiple writers and readers in a database

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. So, for example, while User 1's transaction is inserting and deleting in the customer table, User 2 can begin a transaction that loads data into the employee table, as shown in Figure 10-3. 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 query server or the write server, and read-write users connect to the write server. Read-write users may also connect to query servers, but can only modify:

- Persistent objects in the query server's local store
- Global or local temporary tables

**Figure 10-3: Concurrent insertions to different tables**

Data definition operations on a single table lock out all other readers and writers from that table. See “Locks for DDL operations” on page 486 for details.

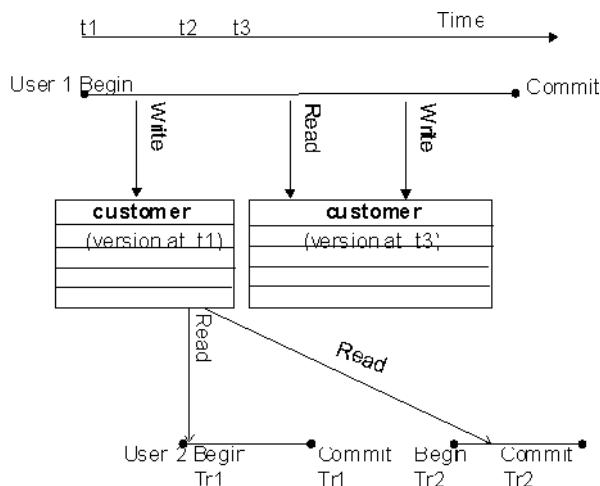
### Transactions use committed data

Committed data results when a write transaction commits. 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.

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 Figure 10-4, 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 10-4: 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 Figure 10-4, 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.

## Timing of commits on read transactions affects versions

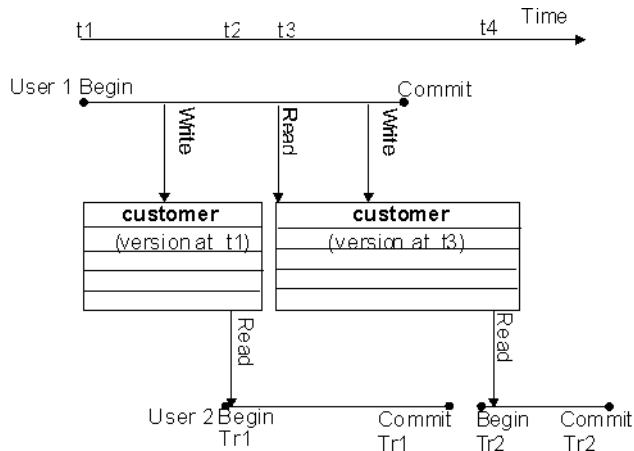
While a read transaction cannot affect what an existing write transaction sees, committing a read transaction does have implications for other transactions.

- 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.

Figure 10-4 on page 482 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.

Figure 10-5 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 10-5: Effect of read transaction committing**



## 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 “Cursors in transactions” for details.

## How Sybase IQ keeps track of 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 drops it from the database. 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.

For information on defining, releasing, and rolling back to savepoints, see “Savepoints within transactions” on page 493.

## 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, a global temporary table is deleted at the next 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.

## 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.

---

**Note** Locks in the Catalog Store are Adaptive Server Anywhere style locks, which work differently. For information on these locks, see the *Adaptive Server Anywhere User's Guide*.

---

## 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.

## 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.
- In an IQ multiplex you can execute CREATE, DROP, and ALTER commands for tables in the Main IQ Store, but only on the write server. To complete DDL operations for the IQ Store, query servers must disconnect users that hold locks on the target database. CREATE, DROP, and ALTER commands are not permitted on query servers, except for tables in the Catalog Store, on temporary tables, and on local IQ Main Stores. (Local IQ Main Stores are only visible to users of that query server.)

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.

See “Versioning of temporary tables” on page 484 for special rules regarding temporary tables.

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.

Users on query servers are not permitted to modify tables in the shared IQ Main Store. If they attempt to do so, they get the error message:

Not permitted on multiplex

Users on query servers may create and modify tables on local IQ Main Stores, but such tables will only be visible to other users of that query server.

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.

Concurrency rules for index creation commands

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.

GRANT, REVOKE, and SET OPTION are not restricted

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 the *Sybase IQ Reference Manual* 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.

## 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

If an attempt to write to a table fails because another transaction holds a lock on that table, you see a message like this one:

```
Cannot open the requested object for write in the
current transaction (TxnID1). Another user has write
access in transaction TxnID2.
```

To identify the user who has that table locked, use the `sp_iqtransaction` procedure. Find TxnID2 in the output of `sp_iqtransaction`, and look for the name of the user in the same row of output.

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

For syntax details and sample output, see “`sp_iqlocks` procedure” in the *Sybase IQ Reference Manual*.

The `sp_iqtransaction` procedure provides more detailed information about transactions.

### Managing lock contention

Some load or query performance issues may be traced to 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. To set these options, see “Using command-line switches” on page 45 and “CACHE\_PARTITIONS option” in the *Sybase IQ Reference Manual*.

---

**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.

---

For an example of managing write lock contention on a table, see “Managing write lock contention on a table” in Chapter 1, “Troubleshooting Hints,” of the *Sybase IQ Troubleshooting and Recovery Guide*.

## 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 statement” section in Chapter 6 of the *Sybase IQ System Administration Guide*.

## **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 Adaptive Server 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.

As discussed in Chapter 1, “Overview of Sybase IQ System Administration,” 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. You can adjust the checkpoint interval with the SET OPTION command; see the *Sybase IQ Reference Manual* for details. You probably do not need to adjust the checkpoint time or issue explicit checkpoints, however. Controlling checkpoints is less important in Sybase IQ than in OLTP database products, because IQ writes the actual data pages after each transaction commits.

For more information on checkpoints in recovery, see “How transaction information aids recovery.”

## 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. For more information on savepoints and rollback, see “Naming and nesting savepoints.”

## **Releasing savepoints**

Once a RELEASE SAVEPOINT statement has been executed or the transaction has ended, 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.

## **Rolling back to a savepoint**

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.

## **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.

## Naming and nesting savepoints

Savepoints can be named and they can be nested. By using named, nested savepoints, you can have 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.

## Rolling back transactions

When you roll back a transaction, you undo all of the operations in that transaction. We say that you are rolling back the database, since you are 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.

For more information on cursors, see “Cursors in transactions.” For more information on rollback to a savepoint, see “Rolling back to a savepoint.”

## **System recovery**

In the event of a system failure or power outage, or when you restart the database server after it has been stopped, 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. If you have trouble starting a server or database, or if users are unable to connect to it, see *Sybase IQ Troubleshooting and Recovery Guide* for information on how to proceed. 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. For details, see the *Sybase IQ Troubleshooting and Recovery Guide*.

## How transaction information aids recovery

Sybase IQ's 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 information about checkpoints and disk space availability are also very small.

The transaction log is deleted:

- Always after a full backup.
- Optionally after incremental backup.
- Always after backup files are restored following media failure, and a new log is started.

The checkpoint information is deleted at the next checkpoint. Information related to particular savepoints is deleted when the savepoint is released or rolled back.

For other concurrency issues relating to backing up and restoring databases, see “Concurrency and backups.”

## Performance implications

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.

For an explanation of how Sybase IQ uses the resources discussed in this section, see Chapter 5, “Managing System Resources” in *Sybase IQ Performance and Tuning Guide*.

## 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. For information on sizes of index types, see “Indexing criteria: disk space usage.”

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, so that you do not lose any data that Sybase IQ might put in 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.

---

## 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. For more information, see “sp\_iqcursorinfo procedure” in Chapter 10, “System Procedures” of the *Sybase IQ Reference Manual*.

## 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.

See “Effect of rollback” on page 495 for what happens to cursors during a rollback of the database.

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.

## 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`.

You can force all cursors to be non-scrolling by setting the option `FORCE_NO_SCROLL_CURSORS` to ON. You may want to use this option to save on temporary storage requirements if you are retrieving very large numbers (millions) of rows.

## 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.

## Cursor command syntax and examples

For more information on using cursors in procedures, including examples of cursor use, see Chapter 8, “Using Procedures and Batches.” For syntax of cursor-related commands, see the *Sybase IQ Reference Manual*.

## Controlling 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. See Chapter 2, “Database Options” in the *Sybase IQ Reference Manual* for details on the LOG\_CURSOR\_OPERATIONS option.

# International Languages and Character Sets

## About this chapter

This chapter describes how to configure your Sybase IQ installation to handle international language issues.

## Contents

Topic	Page
Introduction to international languages and character sets	503
Understanding character sets in software	505
Understanding locales	516
Understanding collations	521
Understanding character set translation	531
Collation internals	534
International language and character set tasks	539
Compatibility issues	547
Performance issues	548

## Introduction to international languages and character sets

This section provides 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.

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.

## Sybase IQ international features

Sybase IQ provides two sets of features that are of particular interest when setting up databases for languages.

- **Collations** You can choose from a wide selection of supplied collations when you create a database. By creating your database with the proper collation, you ensure proper sorting of data.

Whenever the database compares strings, sorts strings, or carries out other string operations such as case conversion, it does so using the collation sequence. The database carries out sorting and string comparison when statements such as the following are executed:

- Queries with an ORDER BY clause.
- Expressions that use string functions, such as LOCATE, SIMILAR, SOUNDEX.
- Conditions using the LIKE keyword.

IQ indexes that hold character data are created based on the database collation. The database also uses collations to identify valid or unique identifiers (column names and so on).

- **Character set translation** You can set up Sybase IQ to convert data between the character set encoding on your server and client systems, thus maintaining the integrity of your data even in mixed character set environments.

Character set translation is provided between client and server, and also by the ODBC driver. The Sybase IQ ODBC driver provides OEM to ANSI character set translation and Unicode support.

## Using the default collation

If you use the default actions when creating an IQ database, your database has the collation ISO\_BINENG. This collation provides optimal performance for IQ databases, but not necessarily the most natural sort order. For more information, see “Performance issues” on page 548.

Note that this differs from Adaptive Server 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 it is not possible to set up your system in this default manner, you need to 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. This chapter provides the information you need to make and implement these decisions.

## Character set questions and answers

The following table identifies where you can find answers to questions.

To answer the question...	Consider reading...
How do I set up my computing environment to treat character sets properly?	“Configuring your character set environment” on page 539
How do I decide which collation to use for my database?	“Understanding collations” on page 521
How are characters represented in software, and Sybase IQ in particular?	“Understanding character sets in software” on page 505
What collations does Sybase IQ provide?	“Supplied and recommended collations” on page 523
How do I ensure that error and informational messages sent from the database server to client applications are sent in the proper language and character set for my application?	“Character translation for database messages” on page 531
I have a different character set on client machines from that in use in the database. How can I get characters to be exchanged properly between client and server?	“Starting a database server using character set translation” on page 544
What character sets can I use for connection strings?	“Connection strings and character sets” on page 532
How do I create a collation that is different from the supplied ones?	“Creating a database with a custom collation” on page 546
How do I change the collation sequence of an existing database	“Changing a database from one collation to another” on page 547

## Understanding character sets in software

This section provides general information about software issues related to international languages and character sets.

## Pieces in the character set puzzle

There are several distinct aspects to character storage and display by computer software:

- Each piece of software works with a **character set**. A character set is a set of symbols, including letters, digits, spaces and other symbols.
- To handle these characters, each piece of software employs a character set **encoding**, in which each character is mapped onto one or more **bytes** of information, typically represented as hexadecimal numbers. This encoding is also called a **code page**.
- Database servers, which sort characters (for example, list names alphabetically), use a **collation**. A collation is a combination of a character encoding (a map between characters and hexadecimal numbers) and a **sort order** for the characters. There may be more than one sort order for each character set; for example, a case-sensitive order and a case-insensitive order, or two languages may sort characters in a different order.
- Characters are printed or displayed on a screen using a **font**, which is a mapping between characters in the character set and their appearance. Fonts are handled by the operating system.
- Operating systems also use a **keyboard mapping** to map keys or key combinations on the keyboard to characters in the character set.

## Language issues in client/server computing

Database users working at client applications may see or access strings from the following sources:

- **Data in the database** Strings and other text data are stored in the database. The database server processes these strings when responding to requests.

For example, the database server may be asked to supply all the last names beginning with a letter ordered less than N in a table. This request requires string comparisons to be carried out, and assumes a character set ordering.

The database server receives strings from client applications as streams of bytes. It associates these bytes with characters according to the database character set. Data in some IQ indexes is stored based on the sort order of the collation.

- **Database server software messages** Applications can cause database errors to be generated. For example, an application may submit a query that references a column that does not exist. In this case, the database server returns a warning or error message. This message is held in a **language resource library**, which is a DLL or shared library called by Sybase IQ.
- **Client application** The client application interface displays text, and internally the client application may process text.
- **Client software messages** The client library uses the same language library as the database server to provide messages to the client application.
- **Operating system** The client operating system has text displayed on its interface, and may also process text.

For a satisfactory working environment, all these sources of text must work together. Loosely speaking, they must all be working in the user's language and/or character set.

## 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.

Upper and lower pages

With few exceptions, characters 0 to 127 are the same for all the single-byte code pages. The mapping for this range of characters is called the **ASCII** character set. It includes the English language alphabet in upper and lower case, as well as common punctuation symbols and the digits. This range is often called the **seven-bit** range (because only seven bits are needed to represent the numbers up to 127) or the **lower** page. The characters from 128 to 256 are called **extended characters**, or **upper** code-page characters, and vary from code page to code page.

Problems with code page compatibility are rare if the only characters used are from the English alphabet, as these are represented in the ASCII portion of each code page (0 to 127). However, if other characters are used, as is generally the case in any non-English environment, there can be problems if the database and the application use different code pages.

**Example**

Suppose a database holding French language strings uses code page 850, and the client operating system uses code page 437. The character À (upper case A grave) is held in the database as character \xB7 (decimal value 183). In code page 437, character \xB7 is a graphical character. The client application receives this byte and the operating system displays it on the screen, the user sees a graphical character instead of an A grave.

Remember that the code page used by the client system determines both the values that are sent to server for each character you enter, and the characters that are displayed when particular values are sent to the client from the server. The code page used by the server system determines how the server interprets values the client sends.

## **ANSI and OEM code pages in Windows**

For Microsoft Windows users, the issue is complicated because there are at least two code pages in use on most PCs. This issue affects both Windows clients and servers.

MS-DOS, as well as character-mode applications (those using the console or “DOS box”) in Windows, use code pages taken from the IBM set. These are called **OEM code pages** (Original Equipment Manufacturer) for historical reasons.

Windows operating systems do not require the line drawing characters that were held in the extended characters of the OEM code pages, so they use a different set of code pages. These pages are based on the ANSI standard and are therefore commonly called **ANSI code pages**.

Sybase IQ supports collations based on both OEM and ANSI code pages.

**Example**

Consider the following situation:

- A PC is running the Windows 98 operating system with ANSI code page 1252.
- The code page for character-mode applications is OEM code page 437.
- Text is held in a database created using the collation corresponding to OEM code page 850.

An uppercase A grave in the database is stored as character 183. This value is displayed as a graphical character in a character-mode application. The same character is displayed as a dot in a Windows application.

For information about choosing a single-byte collation for your database, see “Understanding collations” on page 521.

## Supported code pages

The following table lists code pages supported in Sybase IQ.

Code Page	Description
1252	Windows Latin-1
037	USA, Canada (Bilingual, French), Netherlands, Portugal, Brazil, Australia
273	IBM Austria, Germany
277	IBM Denmark, Norway
278	IBM Finland, Sweden
280	IBM Italy
284	IBM Catalan/Spain, Spanish Latin America
285	IBM United Kingdom, Ireland
297	IBM France
420	IBM Arabic
424	IBM Hebrew
437	MS-DOS United States, Australia, New Zealand, South Africa
500	EBCDIC 500V1
737	PC Greek
775	PC Baltic
838	IBM Thailand extended SBCS
850	MS-DOS Latin-1
852	MS-DOS Latin-2
855	IBM Cyrillic
856	IBM Hebrew
857	IBM Turkish
858	Variant of Cp850 with Euro character
860	MS-DOS Portuguese
861	MS-DOS Icelandic
862	PC Hebrew
863	MS-DOS Canadian French

<b>Code Page</b>	<b>Description</b>
864	PC Arabic
865	MS-DOS Nordic
866	MS-DOS Russian
868	MS-DOS Pakistan
869	IBM Modern Greek
870	IBM Multilingual Latin-2
871	IBM Iceland
874	IBM Thai
875	IBM Greek
918	IBM Pakistan (Urdu)
921	IBM Latvia, Lithuania (AIX, DOS)
922	IBM Estonia (AIX, DOS)
930	Japanese Katakana-Kanji mixed with 4370 UDC, superset of 5026
933	Korean Mixed with 1880 UDC, superset of 5029
935	Simplified Chinese Host mixed with 1880 UDC, superset of 5031
937	Traditional Chinese Host mixed with 6204 UDC, superset of 5033
939	Japanese Latin Kanji mixed with 4370 UDC, superset of 5035
942	IBM OS/2 Japanese, superset of Cp932
942	C Variant of Cp942
943	IBM OS/2 Japanese, superset of Cp932 and Shift-JIS
943	C Variant of Cp943
948	OS/2 Chinese (Taiwan) superset of 938
949	PC Korean
949	C Variant of Cp949
950	PC Chinese (Hong Kong, Taiwan)
964	AIX Chinese (Taiwan)
970	AIX Korean
1006	IBM AIX Pakistan (Urdu)
1025	IBM Multilingual Cyrillic: Bulgaria, Bosnia, Herzegovinia, Macedonia (FYR)
1026	IBM Latin-5, Turkey
1046	IBM Arabic - Windows
1097	IBM Iran (Farsi)/Persian

Code Page	Description
1098	IBM Iran (Farsi)/Persian (PC)
1112	IBM Latvia, Lithuania
1122	IBM Estonia
1123	IBM Ukraine
1124	IBM AIX Ukraine
1140	Variant of Cp037 with Euro character
1141	Variant of Cp273 with Euro character
1142	Variant of Cp277 with Euro character
1143	Variant of Cp278 with Euro character
1144	Variant of Cp280 with Euro character
1145	Variant of Cp284 with Euro character
1146	Variant of Cp285 with Euro character
1147	Variant of Cp297 with Euro character
1148	Variant of Cp500 with Euro character
1149	Variant of Cp871 with Euro character
1250	Windows Eastern European
1251	Windows Cyrillic
1253	Windows Greek
1254	Windows Turkish
1255	Windows Hebrew
1256	Windows Arabic
1257	Windows Baltic
1258	Windows Vietnamese
1381	IBM OS/2, DOS People's Republic of China (PRC)
1383	IBM AIX People's Republic of China (PRC)
33722	IBM-eucJP - Japanese (superset of 5050)
ASCII	American Standard Code for Information Interchange
ISO8859_1	ISO 8859-1, Latin alphabet No. 1
UnicodeBig	Sixteen-bit Unicode Transformation Format, big-endian byte order, with byte-order mark
UnicodeBigUnmarked	Sixteen-bit Unicode Transformation Format, big-endian byte order
UnicodeLittle	Sixteen-bit Unicode Transformation Format, little-endian byte order, with byte-order mark
UnicodeLittleUnmarked	Sixteen-bit Unicode Transformation Format, little-endian byte order
UTF8	Eight-bit Unicode Transformation Format

<b>Code Page</b>	<b>Description</b>
UTF-16	Sixteen-bit Unicode Transformation Format, byte order specified by a mandatory initial byte-order mark
Big5	Big5, Traditional Chinese
Big5_HKSCS	Big5 with Hong Kong extensions, Traditional Chinese
Big5_Solaris	Big5 with seven additional Hanzi ideograph character mappings for the Solaris zh_TW.BIG5 locale
EUC_CN	GB2312, EUC encoding, Simplified Chinese
EUC_JP	JIS X 0201, 0208, 0212, EUC encoding, Japanese
EUC_KR	KS C 5601, EUC encoding, Korean
EUC_TW	CNS11643 (Plane 1-3), EUC encoding, Traditional Chinese
GB18030	Simplified Chinese, PRC standard
GBK	GBK, Simplified Chinese
ISCII91	ISCII91 encoding of Indic scripts
ISO2022CN	ISO 2022 CN, Chinese (conversion to Unicode only)
ISO2022CN_CNS	CNS 11643 in ISO 2022 CN form, Traditional Chinese (conversion from Unicode only)
ISO2022CN_GB	GB 2312 in ISO 2022 CN form, Simplified Chinese (conversion from Unicode only)
ISO2022JP	JIS X 0201, 0208 in ISO 2022 form, Japanese
ISO2022KR	ISO 2022 KR, Korean
ISO8859_2	ISO 8859-2, Latin alphabet No. 2
ISO8859_3	ISO 8859-3, Latin alphabet No. 3
ISO8859_4	ISO 8859-4, Latin alphabet No. 4
ISO8859_5	ISO 8859-5, Latin/Cyrillic alphabet
ISO8859_6	ISO 8859-6, Latin/Arabic alphabet
ISO8859_7	ISO 8859-7, Latin/Greek alphabet
ISO8859_8	ISO 8859-8, Latin/Hebrew alphabet
ISO8859_9	ISO 8859-9, Latin alphabet No. 5
ISO8859_13	ISO 8859-13, Latin alphabet No. 7
ISO8859_15_FDIS	ISO 8859-15, Latin alphabet No. 9
JIS0201	JIS X 0201, Japanese
JIS0208	JIS X 0208, Japanese
JIS0212	JIS X 0208, Japanese
JISAutoDetect	Detects and converts from Shift-JIS, EUC-JP, ISO 2022 JP (conversion to Unicode only)
Johab	Johab, Korean
KOI8_R	KOI8-R, Russian

Code Page	Description
MS874	Windows Thai
MS932	Windows Japanese
MS936	Windows Simplified Chinese
MS949	Windows Korean
MS950	Windows Traditional Chinese
MacArabic	Macintosh Arabic
MacCentralEurope	Macintosh Latin-2
MacCroatian	Macintosh Croatian
MacCyrillic	Macintosh Cyrillic
MacDingbat	Macintosh Dingbat
MacGreek	Macintosh Greek
MacHebrew	Macintosh Hebrew
MacIceland	Macintosh Iceland
MacRoman	Macintosh Roman
MacRomania	Macintosh Romania
MacSymbol	Macintosh Symbol
MacThai	Macintosh Thai
MacTurkish	Macintosh Turkish
MacUkraine	Macintosh Ukraine
SJIS	Shift-JIS, Japanese
TIS620	TIS620, Thai

## Multibyte character sets

Some languages, such as Japanese and Chinese, have many more than 256 characters. These characters cannot all be represented using a single byte, but can be represented in multibyte character sets. In addition, some character sets use the much larger number of characters available in a multibyte representation to represent characters from many languages in a single, more comprehensive, character set.

Multibyte character sets are of two types. Some are **variable width**, in which some characters are single-byte characters, others are double-byte, and so on. Other sets are **fixed width**, in which all characters in the set have the same number of bytes. Sybase IQ supports only variable-width character sets.

**Example**

As an example, characters in the Shift-JIS character set are of either one or two bytes in length. If the value of the first byte is in the range of hexadecimal values from `\x81` to `\x9F` or from `\xE0` to `\xEF` (decimal values 129-159 or 224-239) the character is a two-byte character and the subsequent byte (called a **follow byte**) completes the character. If the first byte is outside this range, the character is a single-byte character and the next byte is the first byte of the following character.

- The properties of any Shift-JIS character can be read from its first byte also. Characters with a first byte in the range `\x09` to `\x0D`, or `\x20`, are space characters.
- Characters in the ranges `\x41` to `\x5A`, `\x61` to `\x7A`, `\x81` to `\x9F` or `\xA1` to `\xEF` are considered to be alphabetic (letters).
- Characters in the range `\x30` to `\x39` are digits.

In building custom collations, you can specify which ranges of values for the first byte signify single- and double-byte (or more) characters, and which specify space, alpha, and digit characters. However, all first bytes of value less than 64 (hex 40) must be single-byte characters, and no follow bytes may have values less than 64. This restriction is satisfied by all known current encodings.

For information on the multibyte character sets, see “Using multibyte collations” on page 530.

## Sorting characters using collations

The database collation sequence includes the notion of alphabetic ordering of letters, and extends it to include all characters in the character set, including digits and space characters.

Associating more than one character with each sort position

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 IQ databases created as of version 12.4.2), 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.

---

**Tip**

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 Müller 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.

## International aspects of case sensitivity

Sybase IQ is **case preserving** and **case insensitive** for identifiers, such as table names and column names. This means that the names are stored in the case in which they are created, but any access to the identifiers is done in a case-insensitive manner.

For example, the names of the system tables are held in upper case (SYSDOMAIN, SYSTABLE, and so on), but access is case insensitive, so that the two following statements are equivalent:

```
SELECT *
FROM systable
SELECT *
FROM SYSTABLE
```

The equivalence of uppercase and lowercase characters is enforced in the collation. There are some collations where particular care may be needed when assuming case insensitivity of identifiers. In particular, Turkish collations have a case-conversion behavior that can cause unexpected and subtle errors. The most common error is that a system object containing a letter i is not found.

For more information on Turkish case behavior, see “Turkish character sets and collations” in the chapter titled “International Languages and Character Sets,” in *Adaptive Server Anywhere Database Administration Guide*.

## Understanding locales

Both the database server and the client library recognize their language and character set environment using a **locale definition**.

### Introduction to locales

The application locale, or client locale, is used by the client library when making requests to the database server, to determine the character set in which results should be returned. If character-set translation is enabled (the default), the database server compares its own locale with the application locale to determine whether character set translation is needed. Different databases on a server may have different locale definitions.

For information on enabling character-set translation, see “Starting a database server using character set translation” on page 544.

The locale consists of the following components:

- **Language**     The language is a two-character string using the ISO-639 standard values: DE for German, FR for French, and so on. Both the database server and the client have language values for their locale.

The database server uses the locale language to determine:

- Which language library to load.

---

**Note** The default collation when you create a Sybase IQ database is ISO\_BINENG. The language on the server has no effect on this default.

---

The client library uses the locale language to determine:

- Which language library to load.
- Which language to request from the database.

For more information, see “Understanding the locale language” on page 517.

- **Character set** The character set is the code page in use. The client and server both have character set values, and they may differ. If they differ, character set translation may be required to enable interoperability.

For machines that use both OEM and ANSI code pages, the ANSI code page is the value used here.

For more information, see “Understanding the locale character set” on page 518.

## Understanding the locale language

The locale language is an indicator of the language being used by the user of the client application, or expected to be used by users of the database server.

For a list of supported locale languages, see “Language label values” on page 517.

For more information about how to find locale settings, see “Determining locale information” on page 540.

The database server determines the language component of the locale as follows:

- 1 It checks the ASLANG environment variable, if it exists.  
For more information, see “ASLANG environment variable” in *Sybase IQ Reference Manual*.
- 2 On Windows, it checks the Adaptive Server Anywhere language registry entry.
- 3 Queries the operating system.
- 4 If the language cannot be determined by the above settings, it defaults to English.

### Language label values

The following table shows the valid language label values, together with the equivalent ISO 639 labels:

Language label	Alternative label	ISO_639 language code
chinese	simpchin	ZH
danish	N/A	DA
czech	N/A	CS
dutch	N/A	NL
french	N/A	FR

Language label	Alternative label	ISO_639 language code
german	N/A	DE
greek	N/A	EL
hebrew	N/A	HE
hungarian	N/A	HU
italian	N/A	IT
japanese	N/A	JA
korean	N/A	KO
lithuanian	N/A	LT
norwegian	norweg	NO
polish	N/A	PL
portuguese	portugue	PT
russian	N/A	RU
spanish	N/A	ES
swedish	N/A	SV
tchinese	tradchin	TW
turkish	N/A	TR
ukrainian	N/A	UK
us_english	english	EN

## Understanding the locale character set

Both application and server locale definitions have a character set. The application uses its character set when requesting character strings from the server. If character set translation is enabled (the default), the database server compares its character set with that of the application to determine whether character set translation is needed.

For a list of available character set labels, see “Character set labels” on page 519.

For more information about how to find locale settings, see “Determining locale information” on page 540.

The client library determines the character set as follows:

- 1 If the connection string specifies a character set, it is used.

For more information, see “CharSet connection parameter [CS]” on page 138.

- 2 Open Client applications check the *locales.dat* file in the Sybase *locales* directory is used.
- 3 Character set information from the operating system is used to determine the locale:
  - On Windows operating systems, use the GetACP system call. This returns the ANSI character set, not the OEM character set.
  - On UNIX, default to ISO8859-1.

The database server determines the character set for a connection as follows:

- 1 The character set specified by the client is used if it is supported.  
For more information, see “CharSet connection parameter [CS]” on page 138.
- 2 The database's character set is used if the client specifies a character set that is not supported.

When a new database is created, the database server determines the character set for the new database as follows.

- 1 A collation is specified in the CREATE DATABASE statement.
- 2 The ASCHARSET environment variable is used if it exists.
- 3 Character set information from the operating system is used to determine the locale.
  - On Windows operating systems, use the GetACP system call. This returns the ANSI character set, not the OEM character set.
  - On UNIX, default to ISO8859-1.
  - On other platforms, use code page 1252.

When creating an IQ database, the default collation of ISO\_BINENG is used if none is explicitly specified.

#### Character set labels

The following table shows the valid character set label values, together with the equivalent IANA labels and a description:

Character set label	IANA label	Description
big5	<N/A>	Traditional Chinese (cf. CP950)
cp437	<N/A>	IBM CP437 - U.S. code set
cp850	<N/A>	IBM CP850 - European code set
cp852	<N/A>	PC Eastern Europe

<b>Character set label</b>	<b>IANA label</b>	<b>Description</b>
cp855	<N/A>	IBM PC Cyrillic
cp856	<N/A>	Alternate Hebrew
cp857	<N/A>	IBM PC Turkish
cp860	<N/A>	PC Portuguese
cp861	<N/A>	PC Icelandic
cp862	<N/A>	PC Hebrew
cp863	<N/A>	IBM PC Canadian French code page
cp864	<N/A>	PC Arabic
cp865	<N/A>	PC Nordic
cp866	<N/A>	PC Russian
cp869	<N/A>	IBM PC Greek
cp874	<N/A>	Microsoft Thai SB code page
cp932	windows-31j	Microsoft CP932 = Win31J-DBCS
cp936	<N/A>	Simplified Chinese
cp949	<N/A>	Korean
cp950	<N/A>	PC (MS) Traditional Chinese
cp1250	<N/A>	MS Windows Eastern European
cp1251	<N/A>	MS Windows Cyrillic
cp1252	<N/A>	MS Windows US (ANSI)
cp1253	<N/A>	MS Windows Greek
cp1254	<N/A>	MS Windows Turkish
cp1255	<N/A>	MS Windows Hebrew
cp1256	<N/A>	MS Windows Arabic
cp1257	<N/A>	MS Windows Baltic
cp1258	<N/A>	MS Windows Vietnamese
deckanji	<N/A>	DEC UNIX JIS encoding
euccns	<N/A>	EUC CNS encoding: Traditional Chinese with extensions
eucgb	<N/A>	EUC GB encoding = Simplified Chinese
eucjis	euc-jp	Sun EUC JIS encoding
eucksc	<N/A>	EUC KSC Korean encoding (cf. CP949)
greek8	<N/A>	HP Greek-8
iso_1	iso_8859-1:1987	ISO 8859-1 Latin-1
iso15	<N/A>	ISO 8859-15 Latin-1 with Euro, etc.
iso88592	iso_8859-2:1987	ISO 8859-2 Latin-2 Eastern Europe
iso88595	iso_8859-5:1988	ISO 8859-5 Latin/Cyrillic

<b>Character set label</b>	<b>IANA label</b>	<b>Description</b>
iso88596	iso_8859-6:1987	ISO 8859-6 Latin/Arabic
iso88597	iso_8859-7:1987	ISO 8859-7 Latin/Greek
iso88598	iso_8859-8:1988	ISO 8859-8 Latin/Hebrew
iso88599	iso_8859-9:1989	ISO 8859-9 Latin-5 Turkish
koi8	<N/A>	KOI-8 Cyrillic
mac	macintosh	Standard Mac coding
mac_cyr	<N/A>	Macintosh Cyrillic
mac_ee	<N/A>	Macintosh Eastern European
macgrk2	<N/A>	Macintosh Greek
macturk	<N/A>	Macintosh Turkish
roman8	hp-rpman8	HP Roman-8
sjis	shift_jis	Shift JIS (no extensions)
tis620	<N/A>	TIS-620 Thai standard
turkish8	<N/A>	HP Turkish-8
utf8	utf-8	UTF-8 treated as a character set

## Understanding collations

This section describes the supplied collations, and provides suggestions as to which collations to use under certain circumstances.

For information on how to create a database with a specific collation, see “Creating a database with a named collation” on page 543.

## Choosing collations

When you create a database, Sybase IQ lets you choose a collation or rely on the default of ISO\_BINENG. In most cases, the default collation is a suitable choice, but you can also explicitly choose a collation to match your needs from the wide selection of supplied collations.

In some cases, Sybase IQ supports more than one collation for a particular language. Different collations vary in the characters they include and in the way they sort special characters, including ligatures (characters consisting of two or more characters joined together) and accented characters. You should choose a collation that uses a character set and sort order that are appropriate for the data in your database.

For more information about sorting of data and international features, see “Sybase IQ international features” on page 503.

Another consideration when you choose a collation is whether to use an ANSI or an OEM collation. OEM collations are suited to character-mode applications (those using the console or command prompt window) in Windows 95/98/Me and Windows NT/2000/XP. ANSI collations are designed for Windows environments, and are recommended in the majority of cases.

For more information about the differences between ANSI and OEM collations, see “ANSI and OEM code pages in Windows” on page 508 and “ANSI or OEM?” on page 526.

## Displaying collations

Each time the database server opens an IQ database, it displays the following collation information:

- The collation (ASA\_Label)
- Case sensitivity (Case)
- Blank padding (Blank Padding) if it was specified when the database was created.

You can also see the collation of your current database by using the `dbcollat` utility to write the collation into a file:

```
dbcollat -c "connection-string" filename
```

For example, you might extract the collation from the `asiqdemo` database as follows:

```
dbcollat -c "uid=DBA;pwd=SQL;eng=myhost_asiqdemo"  
demo_col
```

You can display the contents of `filename` from the File menu in `dbisql`.

To display the version and build number of Adaptive Server Anywhere, use the `-v` option of `dbcollat`. For example:

```
dbcollat -v "uid=DBA;pwd=SQL;eng=myhost_asiqdemo"
Adaptive Server Anywhere Collation Utility Version
9.0.1.1925
```

The preceding output indicates version 9.0.1, build 1925. You can display the same information from dbisql by selecting Help > About.

To see details of any collation that exists on your system, use the -z option of dbcollat. For example, to extract collation 850, you could enter:

```
dbcollat -c "uid=DBA;pwd=SQL;eng=myhost_asiqdemo" -z
850 demo_col
```

## Supplied and recommended collations

The following collations are supplied with Sybase IQ. You can obtain the list of all available collations by typing the following command at the command prompt:

```
SELECT * FROM SYS.SYSCOLLATION
```

Collation label	Description
ISO_BINENG	Binary ordering, English ISO/ASCII 7-bit letter case mappings
874THAIBIN	Code Page 874, Windows Thai
932JPN	Code Page 932, Japanese Shift-JIS with Microsoft extensions
936ZHO	Code Page 936, Simplified Chinese, PRC GBK 2312-80 8-bit encoding
949KOR	Code Page 949, Korean KS C 5601-1987 encoding, Wansung
950ZHO_HK	Code Page 950, Traditional Chinese, Big 5 encoding with HKSCS
950ZHO_TW	Code Page 950, Traditional Chinese, Big 5 encoding
1250LATIN2	Code Page 1250, Windows Latin 2, Central/Eastern European
1250POL	Code Page 1250, Windows Latin 2, Polish
1251CYR	Code Page 1251, Cyrillic
1252LATIN1	Code Page 1252, Windows Latin 1, Western
1252SPA	Code Page 1252, Windows Latin 1, Spanish
1252SWEFIN	Code Page 1252, Windows Latin 1, Swedish/Finnish
1253ELL	Code Page 1253, Windows Greek, ISO8859-7 with extensions
1254TRK	Code Page 1254, Windows Latin 5, Turkish, ISO 8859-9 with extensions

Collation label	Description
1255HEB	Code Page 1255, Windows Hebrew, ISO8859-8 with extensions
1256ARA	Code Page 1256, Windows Arabic, ISO8859-6 with extensions
1257LIT	Code Page 1257, Lithuanian
EUC_JAPAN	Japanese EUC JIS X 0208-1990 and JIS X 0212-1990 Encoding
EUC_CHINA	Simplified Chinese GB 2312-80 Encoding
EUC_TAIWAN	Taiwanese Big 5 Encoding
EUC_KOREA	Korean KS C 5601-1992 Encoding, Johab
ISO_1	ISO8859-1, Latin 1, Western
ISO1LATIN1	ISO8859-1, ISO Latin 1, Western, Latin 1 Ordering
ISO9LATIN1	ISO8859-15, ISO Latin 9, Western, Latin 1 Ordering
UTF8	Unicode Transformation Format-8, 8-bit multibyte encoding for Unicode

The following collations are recommended because they provide the most likely match to the character set being used by the application: they have an appropriate sort order and support for the symbols and accented characters required for each specific language. Some languages do not have a recommended UNIX collation.

Language	Windows collations	UNIX collations
Western European (including English, French, German, Italian, Portuguese, and Spanish)	1252LATIN1	ISO9LATIN1, ISO1LATIN1
Eastern European (including Croatian, Czech, Hungarian, Romanian, Serbian, Slovakian, and Slovenian)	1250LATIN2	
Arabic	1256ARA	
Greek	1253ELL	
Hebrew	1255HEB	
Japanese	932JPN	EUC_JAPAN
Korean	949KOR	EUC_KOREA
Lithuanian	1257LIT	
Polish	1250POL	
Russian and Ukrainian	1251CYR	
Simplified Chinese	936ZHO	EUC_CHINA

Netsca

Language	Windows collations	UNIX collations
Spanish	1252SPA	ISOLATIN1, ISO9LATIN1
Thai	874THAIBIN	874THAIBIN
Traditional Chinese	950ZHO_TW	
Traditional Chinese + HKSCS	950ZHO_HK	
Turkish	1254TRK	

## Alternate collations

Alternate collations are available for compatibility with older versions of Adaptive Server Anywhere, or for special purposes.

Collation label	Type	Description
437LATIN1	OEM	Code Page 437, Latin 1, Western
437ESP	OEM	Code Page 437, Spanish
437SVE	OEM	Code Page 437, Swedish/Finnish
819CYR	ANSI	Code Page 819, Cyrillic
819DAN	ANSI	Code Page 819, Danish
819ELL	ANSI	Code Page 819, Greek
819ESP	ANSI	Code Page 819, Spanish
819ISL	ANSI	Code Page 819, Icelandic
819LATIN1	ANSI	Code Page 819, Latin 1, Western
819LATIN2	ANSI	Code Page 819, Latin 2, Central/Eastern European
819NOR	ANSI	Code Page 819, Norwegian
819RUS	ANSI	Code Page 819, Russian
819SVE	ANSI	Code Page 819, Swedish/Finnish
819TRK	ANSI	Code Page 819, Turkish
850CYR	OEM	Code Page 850, Cyrillic, Western
850DAN	OEM	Code Page 850, Danish
850ELL	OEM	Code Page 850, Greek
850ESP	OEM	Code Page 850, Spanish
850ISL	OEM	Code Page 850, Icelandic
850LATIN1	OEM	Code Page 850, Latin 1, Western
850LATIN2	OEM	Code Page 850, Latin 2, Central/Eastern European
850NOR	OEM	Code Page 850, Norwegian
850RUS	OEM	Code Page 850, Russian

<b>Collation label</b>	<b>Type</b>	<b>Description</b>
850SVE	OEM	Code Page 850, Swedish/Finnish
850TRK	OEM	Code Page 850, Turkish
852LATIN2	OEM	Code Page 852, Latin 2, Central/Eastern European
852CYR	OEM	Code Page 852, Cyrillic
852POL	OEM	Code Page 852, Polish
855CYR	OEM	Code Page 855, Cyrillic
856HEB	OEM	Code Page 856, Hebrew
857TRK	OEM	Code Page 857, Turkish
860LATIN1	OEM	Code Page 860, Latin 1, Western
861ISL	OEM	Code Page 861, Icelandic
862HEB	OEM	Code Page 862, Hebrew
863LATIN1	OEM	Code Page 863, Latin 1, Western
865NOR	OEM	Code Page 865, Norwegian
866RUS	OEM	Code Page 866, Russian
869ELL	OEM	Code Page 869, Greek
920TRK	ANSI	Code Page 920, Turkish, ISO-8859-9
1252DEU	ANSI	Code Page 1251, Windows Specialty German, Umlaut characters not equal
1254TRKALT	ANSI	Code Page 1254, Windows Specialty Turkish, I-dot and I-no-dot are equal

## ANSI or OEM?

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. In most cases, using 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 jConnect driver or the iAnywhere JDBC driver (depending on which method you use) handles character set translation.

## Notes on ANSI collations

The ISO_1 collation	<p>ISO_1 is provided for compatibility with the Adaptive Server Enterprise default ISO_1 collation. The differences are as follows:</p> <ul style="list-style-type: none"><li>• The lower case letter sharp s (\xDF) sorts with the lower case s in Sybase IQ and Adaptive Server Anywhere, but after ss in Adaptive Server Enterprise.</li><li>• The ligatures corresponding to AE and ae (\xC6 and \xE6) sort after A and a respectively in Sybase IQ and Adaptive Server Anywhere, but after AE and ae in Adaptive Server Enterprise.</li></ul>
The 1252LATIN1 collation	<p>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.</p> <p>Windows NT service patch 4 changes the default character set in many locales to a new 1252 character set on which 1252LATIN1 is based. If you have this service patch, you should use this collation.</p> <p>The euro symbol sorts with the other currency symbols.</p>
The ISO1LATIN1 collation	<p>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.</p> <p>If you are not concerned with Adaptive Server Enterprise compatibility, ISO1LATIN1 is generally recommended instead of ISO_1.</p> <p>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.</p>
The ISO9LATIN1 collation	<p>This collation is the same as ISO1LATIN1, but includes the euro currency symbol and the other new characters included in the 1252 LATIN1 collation.</p> <p>If your machine uses the ISO Latin 9 character set, and you are willing to sacrifice some of the optimal performance of ISO_BINENG, then you should use this collation.</p>

## Notes on OEM collations

The following table shows the built-in collations that correspond to OEM code pages. The table and the corresponding collations were derived from several manuals from IBM concerning National Language Support, subject to the restrictions mentioned above. (This table represents the best information available at the time of writing. Due to continuing rapid geopolitical changes, the table may contain names for countries that no longer exist.)

Country	Language	Primary Code Page	Primary Collation	Secondary Code Page	Secondary Collation
Argentina	Spanish	850	850ESP	437	437ESP
Australia	English	437	437LATIN1	850	850LATIN1
Austria	German	850	850LATIN1	437	437LATIN1
Belgium	Belgian Dutch	850	850LATIN1	437	437LATIN1
Belgium	Belgian French	850	850LATIN1	437	437LATIN1
Belarus	Belarussian	855	855CYR		
Brazil	Portuguese	850	850LATIN1	437	437LATIN1
Bulgaria	Bulgarian	855	855CYR	850	850CYR
Canada	Cdn French	850	850LATIN1	863	863LATIN1
Canada	English	437	437LATIN1	850	850LATIN1
Croatia	Croatian	852	852LATIN2	850	850LATIN2
Czech Republic	Czech	852	852LATIN2	850	850LATIN2
Denmark	Danish	850	850DAN		
Finland	Finnish	850	850SVE	437	437SVE
France	French	850	850LATIN1	437	437LATIN1
Germany	German	850	850LATIN1	437	437LATIN1
Greece	Greek	869	869ELL	850	850ELL
Hungary	Hungarian	852	852LATIN2	850	850LATIN2
Iceland	Icelandic	850	850ISL	861	861ISL
Ireland	English	850	850LATIN1	437	437LATIN1
Israel	Hebrew	862	862HEB	856	856HEB
Italy	Italian	850	850LATIN1	437	437LATIN1
Mexico	Spanish	850	850ESP	437	437ESP
Netherlands	Dutch	850	850LATIN1	437	437LATIN1

Country	Language	Primary Code Page	Primary Collation	Secondary Code Page	Secondary Collation
New Zealand	English	437	437LATIN1	850	850LATIN1
Norway	Norwegian	865	865NOR	850	850NOR
Peru	Spanish	850	850ESP	437	437ESP
Poland	Polish	852	852LATIN2	850	850LATIN2
Portugal	Portuguese	850	850LATIN1	860	860LATIN1
Romania	Romanian	852	852LATIN2	850	850LATIN2
Russia	Russian	866	866RUS	850	850RUS
S. Africa	Afrikaans	437	437LATIN1	850	850LATIN1
S. Africa	English	437	437LATIN1	850	850LATIN1
Slovak Republic	Slovakian	852	852LATIN2	850	850LATIN2
Slovenia	Slovenian	852	852LATIN2	850	850LATIN2
Spain	Spanish	850	850ESP	437	437ESP
Sweden	Swedish	850	850SVE	437	437SVE
Switzerland	French	850	850LATIN1	437	437LATIN1
Switzerland	German	850	850LATIN1	437	437LATIN1
Switzerland	Italian	850	850LATIN1	437	437LATIN1
Turkey	Turkish	857	857TRK	850	850TRK
UK	English	850	850LATIN1	437	437LATIN1
USA	English	437	437LATIN1	850	850LATIN1
Venezuela	Spanish	850	850ESP	437	437ESP
Yugoslavia	Macedonian	852	852LATIN2	850	850LATIN2
Yugoslavia	Serbian Cyrillic	855	855CYR	852	852CYR
Yugoslavia	Serbian Latin	852	852LATIN2	850	850LATIN2

## Using multibyte collations

This section describes how multibyte character sets are handled. The description applies to the supported collations and to any multibyte custom collations you may create.

Sybase IQ provides collations using several multibyte character sets.

For a complete listing, see “Understanding collations” on page 521.

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 a utility to convert data files in CP874 format into UTF8, the only Thai language collation supported. For syntax, see the *Sybase IQ Utility Guide*. Before you can load data in the CP874 character set, you must convert 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. 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
```

For more details, see Chapter 5, “SQL Functions” in *Sybase IQ Reference Manual*.

## 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 to be possible. For example, character set translation is possible between EUC-JIS and Shift-JIS character sets, but not between EUC-JIS and OEM code page 850.

This section describes 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 multi-character-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 message is:

Column *column-name* not found

where *column-name* is filled in from the database.

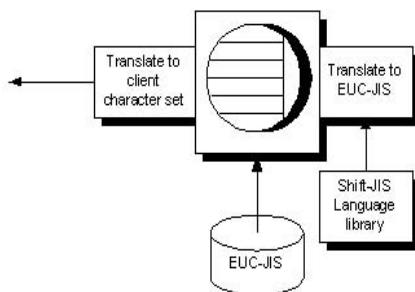
To present these kinds of information to the client application in a consistent manner, even if the database is in a different character set from the language library, the database server automatically translates the characters of the messages so that they match the character set used in the database collation.

❖ **Using character translation for database messages**

- 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.

You must check that the characters of interest to you exist in each character set.

Messages are always translated into the database collation character set, regardless of whether the `-ct` command-line option is used.



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.

## 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:

- 1 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.

- 2 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.

- 3 The DatabaseName (DBN) or DatabaseFile (DBF) parameter is interpreted in the database server character set.
- 4 Once the database is located, the remaining connection parameters are interpreted according to its character set.

## 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. You can turn it off using the -ct- option.

For more information, see Chapter 1, “Running the Database Server” in *Sybase IQ Utility Guide*.

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.

For example, in many environments it is appropriate to use the 1252LATIN1 collation in your database, which corresponds to the Windows code page in many single-byte environments.

- If you are able to 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 the database server is started using the `-ct-` command-line switch.

## Collation internals

This section describes internal technical details of collations, including the file format of collation files.

This section is of particular use if you want to create a database using a custom collation. For information on the steps involved, see “Creating a custom collation” on page 544 and “Creating a database with a custom collation” on page 546.

You can create a database using a collation different from the supplied collations. This section describes how to build databases using such a **custom collation**.

In building multibyte custom collations, you can specify which ranges of values for the first byte signify single- and double-byte (or more) characters, and which specify space, alpha, and digit characters. However, all first bytes of value less than `\x40` must be single-byte characters, and no follow bytes may have values less than `\x40`. This restriction is satisfied by all supported encodings.

Collation files may include the following elements:

- Comment lines, which are ignored by the database.
- A title line.
- A Collation section.

- An Encodings section.
- A Properties section.

## Comment lines

In the collation file, spaces are generally ignored. Comment lines start with either the percent sign (%) or two dashes (--).

## The title line

The first non-comment line must be of the form:

Collation *collation\_label* (*collation\_name*) (*ase\_charset*) (*ase\_so\_sensitive*)  
(*ase\_so\_insensitive*) (*java\_charset*)

### Argument descriptions

Item	Description
Collation	A required keyword.
<i>label</i>	The collation label, which appears in the system tables as SYS.SYSCOLLATION.collation_label and SYS.SYSINFO.default_collation. The label must contain no more than 10 characters.
<i>collation_name</i>	A text description of the collation, usually describing the character set and ordering of the collation.
<i>ase_charset</i>	The Adaptive Server Enterprise name of a character set matching the character set of this collation. This name is used to make character set mappings when server character set translation is enabled (the default).
<i>ase_so_sensitive</i>	The name of an Open Client or Adaptive Server Enterprise case sensitive collation matching this collation.
<i>ase_so_insensitive</i>	The name of an Open Client or Adaptive Server Enterprise case insensitive collation matching this collation.
<i>java_charset</i>	The name of a Java character set matching the character set of this collation. This name is used when converting character data between the Java virtual machine and the database.

For example, the 932JPN collation file contains the following collation line, with label 932JPN and name (Code Page 932, Japanese Shift-JIS with Microsoft extensions):

Collation 932JPN (Code Page 932, Japanese Shift-JIS with

Microsoft extensions) (cp932) (bin\_cp932) (bin\_cp932)  
(SJIS)

## The collation section

After the title line, each non-comment line describes one position in the collation. The ordering of the lines determines the sort ordering used by the database, and determines the result of comparisons. Characters on lines appearing higher in the file (closer to the beginning) sort before characters that appear later.

The form of each line in the sequence is:

[*sort-position*] : *character* [, *character*] ...]

or

[*sort-position*] : *character* [*lowercase* *uppercase*]

### Argument descriptions

Argument	Description
<i>sort-position</i>	Optional. Specifies the position at which the characters on that line will sort. Smaller numbers represent a lesser value, so will sort closer to the beginning of the sorted set. Typically, the sort-position is omitted, and the characters sort immediately following the characters from the previous sort position.
<i>character</i>	The character whose sort-position is being specified.
<i>lowercase</i>	Optional. Specifies the lowercase equivalent of the character. If not specified, the character has no lowercase equivalent.
<i>uppercase</i>	Optional. Specifies the uppercase equivalent of the character. If not specified, the character has no uppercase equivalent.

Multiple characters may appear on one line, separated by commas (,). In this case, these characters are sorted and compared as if they were the same character.

### Specifying character and sort-position

Each character and sort position is specified in one of the following ways:

Specification	Description
\dnnn	Decimal number, using digits 0-9 (such as \d001)
\xhh	Hexadecimal number, using digits 0-9 and letters a-f or A-F (such as \xB4)
'c'	Any character in place of c (such as ',')

Specification	Description
c	Any character other than quote ('), backslash (\), colon (:), or comma (,). These characters must use one of the previous forms.

The following are some sample lines for a collation:

```
% Sort some special characters at the beginning:
: '
: _
: \xF2
: \xEE
: \xF0
: -
: ', '
: ;
: ':'
: !
% Sort some letters in alphabetical order
: A a A
: a a A
: B b B
: b b B
% Sort some E's from code page 850,
% including some accented extended characters:
: e e E, \x82 \x82 \x90, \x8A \x8A \xD4
: E e E, \x90 \x82 \x90, \xD4 \x8A \xD4
```

#### Other syntax notes

For databases using case-insensitive sorting and comparison (that is, CASE IGNORE was specified when the database was created), the lowercase and uppercase mappings are used to find the lowercase and uppercase characters that will be sorted together.

**Note** When a database is created with CASE IGNORE, queries may return data in either upper or lower case, depending on the type of index the optimizer chose to use. You can return all upper case data in such a situation by using this command:

```
SET TEMPORARY OPTION AGGREGATION_PREFERENCE=-2
```

For multibyte character sets, the first byte of a character is listed in the collation sequence, and all characters with the same first byte are sorted together, and ordered according to the value of the following bytes. For example, the following is part of the Shift-JIS collation file:

```
:  \xfcb
:  \xfcc
:  \xfcd
```

In this collation, all characters with first byte \xfc come after all characters with first byte \xfb and before all characters with first byte \xfd. The two-byte character \xfc \x01 would be ordered before the two-byte character \xfc \x02.

Any characters omitted from the collation are added to the end of the collation. The tool that processes the collation file issues a warning.

## The Encodings section

The Encodings section is optional, and follows the collation sequence. It is not useful for single-byte character sets.

The Encodings section lists which characters are lead-bytes, for multibyte character sets, and what are valid follow-bytes.

For example, the Shift-JIS Encodings section is as follows:

```
Encodings:
[ \x00-\x80, \xa0-\xdff, \xf0-\xff ]
[ \x81-\x9f, \xe0-\xef ] [ \x40-\x7e, \x80-\xfc ]
```

The first line following the section title lists valid single-byte characters. The square brackets enclose a comma-separated list of ranges. Each range is listed as a hyphen-separated pair of values. In the Shift-JIS collation, values \x00 to \x80 are valid single-byte characters, but \x81 is not a valid single-byte character.

The second line following the section title lists valid multibyte characters. Any combination of one byte from the second line followed by one byte from the first is a valid character. Therefore \x81\x40 is a valid double-byte character, but \x81 \x00 is not.

## The Properties section

The Properties section is optional, and follows the Encodings section.

If a Properties section is supplied, an Encodings section must be supplied also.

The Properties section lists values for the first-byte of each character that represent alphabetic characters, digits, or spaces.

The Shift-JIS Properties section is as follows:

```
Properties:  
space: [\x09-\x0d, \x20]  
digit: [\x30-\x39]  
alpha: [\x41-\x5a, \x61-\x7a, \x81-\x9f, \xe0-\xef]
```

This indicates that characters with first bytes \x09 to \x0d, as well as \x20, are to be treated as space characters, digits are found in the range \x30 to \x39 inclusive, and alphabetic characters in the four ranges \x41-\x5a, \x61-\x7a, \x81-\x9f, and \xe0-\xef.

## International language and character set tasks

This section groups together the tasks associated with international language and character set issues.

### Finding the default collation

If you do not explicitly specify a collation when creating a database, a default collation is used. For IQ databases, the default collation is always ISO\_BINENG.

### Configuring your character set environment

This section describes how to set up your computing environment so that character set issues are handled properly. If you set your locale environments properly, then you do not need to turn on character set translation between client and server.

#### ❖ Configuring your character set environment

- 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.

For how to find locale information, see “Determining locale information” on page 540.

- 2 Decide whether the locale settings are appropriate for your environment.  
For more information, see “Understanding collations” on page 521.
- 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.  
For more information, see “Avoiding character-set translation” on page 533.
- 4 Set locales on each of the machines in the environment to these values.  
For more information, see “Setting locales” on page 541.
- 5 Create your database using the default collation. If the default collation does not match your needs, create a database using a named collation.  
For more information, see “Creating a database with a named collation” on page 543.

When choosing the collation for your database, consider the following:

- 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.
- There is a performance cost, as well as extra complexity in system configuration, when you use character set translation. Choose a collation that avoids the need for 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 12.7 or higher. You can turn off character set translation using the -ct- option.

For information, see “Avoiding character-set translation” on page 533.

## Determining locale information

You can determine locale information using system functions.

For a complete list, see the *Sybase IQ Reference Manual*.

❖ **Determining the locale of a database server**

- 1 Start DBISQL, and connect to a database server.
- 2 Execute the following statement to determine the database server character set:

```
SELECT PROPERTY( 'CharSet' )
```

The query returns one of the supported character sets listed in “Character set labels” on page 519.

- 3 Execute the following statement to determine the database server language:

```
SELECT PROPERTY( 'Language' )
```

The query returns one of the supported languages listed in “Language label values” on page 517.

- 4 Execute the following statement if you need to determine a good alternative to the default collation, ISO\_BINENG:

```
SELECT PROPERTY( 'DefaultCollation' )
```

The query returns one of the collations listed in “Supplied and recommended collations” on page 523.

Notes

To obtain client locale information, connect to a database server running on your current machine.

To obtain the character set for an individual database, execute the following statement:

```
SELECT DB_PROPERTY( 'CharSet' )
```

## Setting locales

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.

❖ **Setting the Sybase IQ locale on a computer**

- 1 If the default locale is appropriate for your needs, you do not need to take any action.  
To find out the default locale of your operating system, see “Determining locale information” on page 540.

- 2 If you need to change the locale, you can set either or both of the ASLANG and ASCHARSET environment variables:

```
ASCHARSET=charset;ASLANG=language_code
```

where *charset* is a character set label from the list of character set labels in “Understanding the locale character set” on page 518 and *language\_code* is a language label from the list of language label values in “Understanding the locale language” on page 517.

For information on how to set environment variables on different operating systems, see the *Sybase IQ Reference Manual*.

❖ **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 = american.iso99591, us_english, iso_1
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
```

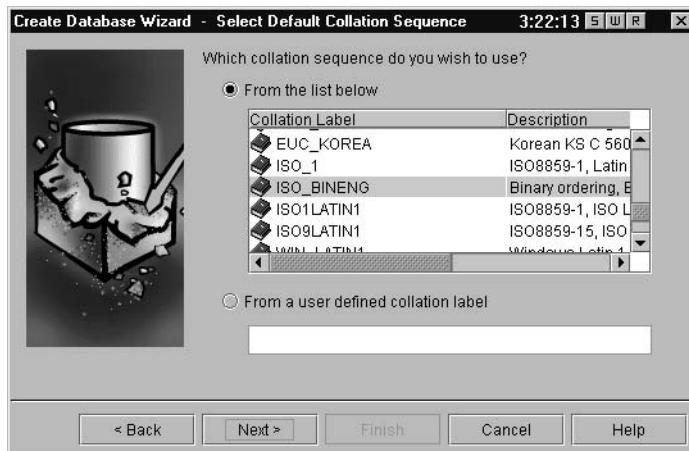
Sybase IQ loads the localization information when it executes the INSERT...LOCATION statement.

## 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.

### ❖ Specifying a database collation when creating a database (Sybase Central)

- You can use the Create Database wizard in Sybase Central to create a database. The wizard has a Collation Sequence page where you choose a collation from a list.



You can also see the name of your current collation in Sybase Central. Right-click on the database whose collation you need. In the dropdown menu select Properties, and then click the Extended Information tab.

### ❖ Specifying a database collation when creating a database (SQL)

- 1 List the supplied collation sequences:

```
SELECT * FROM SYS.SYSCOLLATIONMAPPINGS
```

The first column of the list is the collation label, which you supply when creating the database.

437LATIN1	Code Page 437, Latin 1, Western
437ESP	Code Page 437, Spanish
437SVE	Code Page 437, Swedish/Finnish
819CYR	Code Page 819, Cyrillic
819DAN	Code Page 819, Danish
...	

- 2 Use the CREATE DATABASE statement to create a database. The following statement creates a database with a Greek collation for Windows:

```
CREATE DATABASE 'mydb.db'
COLLATION '1253ELL'
IQ SIZE 100
IQ PATH 'myiq.iq'
```

## Starting a database server using character set translation

Character set translation takes place if the client and server locales are different. Character set translation is enabled by default in Sybase IQ servers version 12.7. You can turn character set conversion on and off explicitly on the database server command line.

### ❖ Enabling character set translation on a database server

- Start the database server using the -ct+ command-line option. For example:

```
start_asiq -ct+ asiqdemo.db
```

### ❖ Disabling character set translation on a database server:

- Start the database server using the -ct- command-line option. For example:

```
start_asiq -ct- asiqdemo.db
```

## Creating a custom collation

If none of the supplied collations meet your needs, you can modify a supplied collation to create a **custom collation**. You can then use this custom collation when creating a database.

For a list of supplied collations, see “Supplied and recommended collations” on page 523.

### ❖ Creating a custom collation

- 1 Decide on a starting collation.

You should choose a collation as close as possible to the one you want to create as a starting point for your custom collation.

For a listing of recommended collations, see “Understanding collations” on page 521.

- 2 Create a custom collation file.

You do this using the Collation utility. The output is a collation file.

For example, the following statement extracts the 1252LATIN1 collation into a file named *mycustomcol.col*:

```
dbcollat -z 1252LATIN1 mycustomcol.col
```

- 3 Edit the custom collation file.

Open the collation file (in this case *mycustomcol.col*) in a text editor.

- 4 Change the collation label.

```
Collation 1252LATIN1 (Code Page 1252, Windows Latin  
1, Western) (cp1252) (dictionary_iso_1)  
(nocase_iso_1) (Cp1252)
```

to

```
Collation MyOrdering (Code Page 1252, My Company  
Ordering) (cp1252) (dictionary_iso_1)  
(nocase_iso_1) (Cp1252)
```

The other entries on this line relate the Sybase IQ collation label to the names that Java and the Sybase TDS interface give to the same collation information. If you are not using these interfaces you do not need to alter these entries.

The Collation line takes the following form:

```
Collation collation_label (collation_name)  
(ase_charset) (ase_so_sensitive)  
(ase_so_insensitive) (java_charset)
```

where character set label (*ase\_charset*) and the two sort-order labels (*ase\_so\_sensitive* and *ase\_so\_insensitive*) state which Open Client character set and sort order is the closest to the current collation. The *java\_charset* label is the closest character set known to Java.

You should edit this line to provide a new label. The label you need to change is the *collation\_label*: in the example in step two, it is 1252LATIN1.

- 5 Change the collation definition.

Make the changes you wish in the custom collation file to define your new collation.

For information on the collation file contents and format, see “Collation internals” on page 534.

6 Convert the file to a SQL script.

You do this using the *dbcollat* command-line utility using the *-d* switch.

For example, the following command line creates the *mycustom.SQL* file from the *mycustomcol.col* collation file:

```
dbcollat -d mycustomcol.col mycustom.SQL
```

7 Add the SQL script to the script in your installation.

The script used when creating databases is held in the *scripts* subdirectory of your Sybase IQ installation directory. Append the contents of *mycustom.SQL* to end of *custom.SQL*.

The new collation is now in place, and can be used when creating databases.

8 Restart the database.

Stop and restart the database server in order for it to recognize the new collations and insert them into system tables *SYSCOLLATION* and *SYSCOLLATIONMAPPINGS*.

## Creating a database with a custom collation

If none of the supplied collations meet your needs, you can create a database using a custom collation. The custom collation is used in indexes and any string comparisons.

❖ **Creating a database with a custom collation**

1 Create a custom collation.

You must have a custom collation in place to use when creating a database.

For instructions on how to create custom collations, see “Creating a custom collation” on page 544.

2 Create the new database.

Use the *CREATE DATABASE* statement or Sybase Central, specifying the name of your custom collation.

For example, the following statement creates a database named *newcol.db* using the custom collation sequence *newcol*.

```
CREATE DATABASE 'newcol.db'
COLLATION 'newcol'
IQ PATH 'newcol.iq'
```

## Changing a database from one collation to another

Changing your database from one collation to another may be a good idea for any number of reasons. It can be especially useful, for example, to:

- Avoid character set translation across your setup.
- Match the characters in your database to the collation in your database. Using the same character set defined in your database is especially important when sorting.
- Use a different character set. You may, for example, want to move from an OEM character set to a Windows character set.

Simply modifying the collation in an existing database would invalidate indexes for that database, and is not permitted. To change the collation for a database, you must rebuild the database. Rebuilding creates a new database with new settings (including collation settings). You will then need to load the data from a flat file or using `INSERT ..LOCATION`.

## Compatibility issues

Prior to version 12.0, Sybase IQ always used the ASCII sort order, which sorts uppercase characters before lowercase. As of version 12.4.2, 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 non-binary one.

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 non-binary 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 you are willing to sacrifice some degree of performance, use the collation ISO\_1 instead of the default, ISO\_BINENG.

---

**Note** By default, password case sensitivity follows case sensitivity of the database. When your database uses the CASE RESPECT option (the default), passwords are also case sensitive by default. You can override this default using the PASSWORD CASE IGNORE clause of CREATE DATABASE, which makes passwords case insensitive regardless of the database setting.

When passwords are case sensitive, you must enter the password as it has been defined, and the DBA's default password must be entered in uppercase. (The user ID is unaffected by the CASE RESPECT option.) For example, you could enter the CONNECT statement as follows:

```
connect database dbasiq user dba identified by SQL
```

You would not be able to connect if you entered this statement as:

```
connect database dbasiq user dba identified by sq1
```

---

# Managing User IDs and Permissions

## About this chapter

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.

## Contents

Topic	Page
An overview of database permissions	549
Managing IQ user accounts and connections	556
Multiplex login management	561
Utility database server security	564
Managing individual user IDs and permissions	566
Managing groups	574
Database object names and prefixes	579
Using views and procedures for extra security	581
How user permissions are assessed	584
Managing the resources connections use	584
Users and permissions in the system tables	586

## An overview of database permissions

Proper management of user IDs and permissions is essential in a data warehouse. It allows users to carry out their jobs 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 for database users, and to display the current permissions of users.

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. For details, see “Message logging” on page 23.

## DBA authority overview

When a database is created, a single usable user ID is created. This first user ID is DBA and the password is initially set to SQL. 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 needs to 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 chapter, 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.

See the syntax of the commands for creating database objects, Chapter 3, “SQL Language Elements,” in *Sybase IQ Reference Manual*.

**DBA user ID in case sensitive databases**

User IDs and passwords are actually objects in the database. Password case sensitivity follows that of the data by default. For this reason, you must enter the password for the user ID DBA (SQL by default) in uppercase if:

- Your database was created with the CASE RESPECT parameter (but not with the PASSWORD CASE IGNORE parameter), or
- Your database was created with the PASSWORD CASE RESPECT parameter

For case insensitive databases (CASE IGNORE or PASSWORD CASE IGNORE) you can enter this user ID and password in either uppercase or lowercase.

## RESOURCE authority overview

RESOURCE authority is the permission to create database objects, such as tables, views, and stored procedures. Resource authority may be granted only by the DBA to other users.

## Ownership permissions overview

The creator of a database object becomes the owner of that object. Ownership of a database object carries with it permissions to carry out actions on that object. These 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. A use for this ability is discussed in “Groups without passwords”. Despite this possibility, this chapter refers interchangeably to the owner and creator of database objects.

## 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.

## Multiplex permissions overview

In Sybase IQ 12.7, the default behavior is that only write servers can:

- Modify TABLE permissions (select, insert, delete, update, references and alter) on tables owned by the write server.

- Modify EXECUTE permission on stored procedures and functions owned by the write server.
- Execute GRANT/REVOKE of DBA, RESOURCE, GROUP, or MEMBERSHIP.
- Create a user (GRANT CONNECT TO... where the user does not exist).
- Drop a user (REVOKE CONNECT FROM...).

The following objects, when created on a write server, are owned by that server and cannot be dropped or altered on a query server:

- Tables
- Views
- Indexes
- Data types
- Messages
- Constraints
- Procedures
- Functions
- Comments

The following, when created, altered, or dropped on a write server, propagate to query servers:

- IQ base tables, IQ global temporary tables, and indexes on either IQ base or global temporary tables
- IQ referential integrity constraints and IQ check constraints
- User-defined data types (domains)
- Messages
- Users and groups. For more information, see “Multiplex login management” on page 561.
- Permissions
- Views
- Comments
- Stored procedures and functions

The following are permitted on query servers:

- CREATE, ALTER, and DELETE EVENT
- Changing a user password via GRANT CONNECT TO... IDENTIFIED BY ... when the user already exists

The database options MPX\_GLOBAL\_TABLE\_PRIV and MPX\_LOCAL\_SPEC\_PRIV let you override the permission restrictions. For details, see “MPX\_GLOBAL\_TABLE\_PRIV option” and “MPX\_LOCAL\_SPEC\_PRIV option” in *Sybase IQ Reference Manual*.

## Setting multiplex permissions

Sybase strongly recommends that you create users, domains and messages on the write server only; otherwise static collisions could lead to unintended results.

User names on the write server are global. For example, suppose that the DBA creates a user name chris for user Christopher Jones that exists on a query server only and owns objects on that query server.

Subsequently, the user chris is created for user Christine Smith on the write server. Now there is only one user chris, who is Christine Smith, and Christine Smith now owns objects on the query server formally owned by Christopher Jones.

A similar problem exists for domains and messages because both have a flat global namespace. Even though they are owned by a user, there is no way to qualify the name with the owner name.

After a static collision, the original query server message or domain is renamed. However, any query server object that references the original query server message will now reference the new message. A query server object that references the original query server domain will either reference the original or the new domain, depending on whether it references it in a definition (e.g. a table) or dynamically (e.g. in a stored procedure).

Sometimes it may be best to set permissions differently on query servers. Consider these factors:

- Synchronizing copies user and permission definitions from the write server. The stored procedure `sp_mpvcfg_servername` (where `servername` is the query server name) may be used to reset the query server values to the desired state as part of the synchronize. If the query server has an IQ Local Store, the user privileges will be re-established as part of the synchronize. You can use the `-iqlocalreplay` server switch to override this.

- In the event of a disaster, the Catalog Store of a query server may be used to recreate the write server. In that case, differences on the query server leave the new write server in a different state from before the disaster.
- Setting permission on the query server is not permanent. The query server will continue to see GRANT and REVOKE commands from the write server. If the write server resets the permission, the change propagates to the query server.

## Server command-line permission options

The database server startup command `start_asiq` has options that set the permission level of some database and server functions. Table 12-1 lists these options.

**Table 12-1: Startup options affecting permissions**

Option	Description	Allowed values	Default
<code>-gd level</code>	Set permission required to start the database	DBA, ALL, NONE	DBA
<code>-gk level</code>	Set permission required to stop the server	DBA, ALL, NONE	DBA
<code>-gl level</code>	Set permission required to load data	DBA, ALL, NONE	ALL for servers started with <code>start_asiq</code> ; DBA for other servers
<code>-gu level</code>	Set permission required to execute utility commands, for example <code>CREATE DATABASE</code> and <code>DROP DATABASE</code>	DBA, ALL, NONE, UTILITY_DB	ALL

See “Controlling permissions from the command line” on page 53 and Chapter 1, “Running the Database Server” in the *Sybase IQ Utility Guide* for more details on these options and the permission level values and defaults.

## Managing IQ user accounts and connections

The Sybase IQ Login Management facility helps you manage users and connections to a database. There are two ways to use the facility:

- Sybase Central property sheets listed in Table 12-2.
- System procedures listed in Table 12-3.

DBAs can add or drop users and control connections by:

- Limiting the number of active logins for the database
- Limiting the number of active logins for a single user
- Locking out a user
- Setting user password expirations

You can perform many of these Login Management functions on a multiplex write server and propagate them automatically to query servers in the multiplex. For details, see “Multiplex login management” on page 561.

To perform these functions in Sybase Central, use the property sheets listed in Table 12-2. For detailed descriptions of property sheets, see the online help for the Sybase IQ plug-in.

**Table 12-2: Login management in Sybase Central**

Management function	Property sheet	Procedure
Enable Login Management	Database Property Sheet	Right-click database name and choose Properties. Click the Login Management tab.
Manage global server settings for maximum connections or lockout status.	Multiplex Server Property Sheet	Right-click server name and choose Properties. Click the Login Management tab.
Manage per server connection limits, number of connections allowed, expired accounts, and expiration dates. Overrides global server settings.	Database Property Sheet	Right-click database name and choose Properties. Click the Login Management tab.
Manage login controls for given user on any/all servers in the multiplex. Overrides server-specific settings.	User Property Sheet	Right-click user name and choose Properties. Click the Login Management tab. For more information, see “Adding a user to the database” in <i>Introduction to Sybase IQ</i> .

Table 12-3 lists the procedure you call to perform each Sybase IQ Login Management function. DBA authority is required to run all procedures except `sp_iqpassword`. All users can run `sp_iqpassword` to change their own passwords.

**Table 12-3: Stored procedures for login management**

<b>Call this stored procedure...</b>	<b>To perform this task...</b>
<code>sp_iqaddlogin</code>	Add users and define their password, number of concurrent connections, and password expiration
<code>sp_iqdroplogin</code>	Drop users
<code>sp_iqlistexpiredpasswords</code>	List users whose passwords have expired
<code>sp_iqlistlockedusers</code>	List users who are locked out of the database
<code>sp_iqlistpasswordexpirations</code>	List password expiration information for all users
<code>sp_iqlocklogin</code>	Lock a user account so that the user cannot connect to the database
<code>sp_iqmodifyadmin</code>	Enable Sybase IQ Login Management, or set database defaults for active user or database connections or password expirations
<code>sp_iqmodifylogin</code>	Modify the number of concurrent connections or password expiration for one or all users
<code>sp_iqpassword</code>	Modify a user's password. Users can modify their own password. DBAs can modify any password.

Enabling Sybase IQ Login Management updates the system tables with changes made since Login Management was last enabled.

Sybase IQ Login Management requires that the LOGIN\_PROCEDURE database option be set to `dba.sp_iq_process_login`. With this option setting, when Sybase IQ Login Management is enabled, each time a user tries to connect to the database, the `sp_iq_process_login` procedure executes. This procedure determines whether the user is allowed to connect. It checks that the user is not locked out, that the maximum number of connections for that user and for the database are not exceeded, and that the user's password has not expired. It then allows login to proceed or sends an error message if any login condition is not met.

When Sybase IQ Login Management is disabled, user login proceeds without these checks.

While you can also use the GRANT CONNECT and REVOKE CONNECT commands or Sybase Central to add or drop users, you cannot manage logins by those users with the Sybase IQ Login Management stored procedures, unless you also add them with `sp_iqaddlogin`. When you enable Sybase IQ Login Management, existing users are automatically added to the system table used by the Sybase IQ Login Management procedures.

For more information, see the *Sybase IQ Reference Manual*:

- For details of Sybase IQ Login Management procedures, see Chapter 10, “System Procedures”
- For use of the LOGIN\_PROCEDURE option sp\_iq\_process\_login, see Chapter 2, “Database Options”
- For system tables used in Sybase IQ Login Management, see Chapter 9, “System Tables”

**Example**

The following example shows how you can prevent a user from connecting after five failed login attempts. This example uses a ConnectFailed event handler to total failed connect attempts.

The totals are stored in table *dba.event\_table*. The first time a user fails a connection attempt, a row will be inserted to *dba.event\_table* for that user. On subsequent failed connections, the count in that user's row will be updated. A ConnectFailed event cannot prevent a user from continuing to try to connect. The login procedure must be used to deny access if the allowed number of failed login attempts has been exceeded. This example will keep track of failed login attempts for all users including those who may be database administrators.

First, create the table to hold the user information and insert a row for each user.

```
create table dba.event_table ( username char(128) not
null,
                                failed_login_attempts integer,
                                primary key (username) );
```

Create the event handler that will increment the number of failed login attempts.

```
create event ev_badlogin type ConnectFailed handler
begin
    declare uid char(128);
    declare xx integer;
    set uid = event_parameter('User');

    if exists (select * from dba.event_table
               where ucacase(uid) = ucacase(username)) then

        -- The user is already in the table.

        update event_table set failed_login_attempts =
            failed_login_attempts+1
            where ucacase(username) = ucacase(uid);

    else
```

```
-- Insert the user for the first time/
insert dba.event_table values (ucase(uid), 1);
end if;

select failed_login_attempts into xx from
dba.event_table
where ucase(username)=ucase(uid)

-- It's not possible to stop the user from attempting
-- to connect after 5 tries. Instead, send a message
-- to the server console to notify the database
-- administrator that a user has exceeded the
-- allowable connect attempts.

if xx > 5 then
raiserror 17001 uid + ' has had more than 5
failed login attempts.';

end if;

end
```

The following is the login procedure.

```
Create procedure dba.check_logins()
begin
    declare xx integer;
    declare uid char(128);
    -- See if the connected user is in the event_table
    -- and proceed accordingly.

    if exists (select * from dba.event_table
    where ucase(username) = ucase(current user))
    then
        select failed_login_attempts into xx
        from dba.event_table
        where ucase(username) = ucase(current user);
        if ( xx >= 5 ) then
            raiserror 17010 current user + ' has been
            locked out by the Database Administrator.';

    else
        -- The user has connected.
        -- Remove the user's row from dba.event_table and ---
        call the default login procedure for the database.
```

```

set uid=current user;
delete DBA.event_table
    where ucname(username) = ucname(uid);
call sp_login_environment();
end if;
else
-- The user is not in dba.event_table, but has
connected.
-- Call the default login procedure for the database.
call sp_login_environment();
end if;
end;

```

To enable all users to run the login stored procedure and set the login procedure option, enter the following.

```

grant execute on dba.check_logins to PUBLIC;
set option PUBLIC.Login_Procedure = 'dba.check_logins';

```

To enable a user that has exceeded the number of allowed failed connection attempts to connect, a database administrator must delete the row for that user from dba.event\_table.

## Multiplex login management

With some exceptions, you can perform login management operations on any server in a multiplex and propagate them to other servers.

Table 12-4 shows operations that you can propagate.

**Table 12-4: Scope of multiplex login management**

Operation	Propagates From	Scope
Enable/disable login management	Write or query server	Multiplex
Add a user (sp_iqaddlogin or GRANT CONNECT)	Write server only <sup>1</sup>	Multiplex
Drop a user (sp_iqdroplogin or REVOKE CONNECT)	Write server only <sup>1</sup>	Multiplex
Change a user password	Write or query server	Multiplex
Modify password expiration	Write or query server	Multiplex
Lock user out of multiplex	Write or query server	Multiplex
Lock user out of a server <sup>2</sup>	Write or query server	Server
Set per-user connect limit <sup>3</sup>	Write or query server	Multiplex
Set global per-server connect limit <sup>2</sup>	Write or query server	Multiplex
Set per-server connect limit	Write or query server	Server

Operation	Propagates From	Scope
Set default password expiration	Write or query server	Multiplex
Set password expiration warning period	Write or query server	Multiplex

1. By default, disallowed on query server. Allowed only if database option MPX\_LOCAL\_SPEC\_PRIV is set to the appropriate numeric value. For details, see “MPX\_LOCAL\_SPEC\_PRIV option” in *Sybase IQ Reference Manual*. In this case, the query server allows the operation but does not propagate it. Changes are lost when the query server is synchronized.

2. The user locked setting and the per-user connection limit on a particular query server persist after synchronizing, unless SQL Remote was not running. Sybase recommends that you perform these changes on a write server to avoid this situation.

3. The per-user connection limit applies to each server in a multiplex. For example, if the limit is 10, the user may have 10 connections to one query server and another 10 to another query server.

DBA privileges are required for all of the operations in the table except changing a user password. Users can change their own passwords.

Table 12-3 on page 558 lists procedures for managing logins.

Propagation of login management operations requires:

- An active write server.
- Operational and active SQL Remote processes.
- Servers upgraded to Sybase IQ 12.7.

If the write server or SQL Remote processes are temporarily inactive, settings propagate only when they become active.

For example, you start a query server in single-node mode, perform login management operations, then restart it as a write server, all login activities propagate to query servers when they are synchronized. If, however, you restart the server as a query server instead, the write server overwrites all login changes at synchronization.

Any changes, such as a reset password, made on the query server when the write server or SQL Remote processes are temporarily down propagate once they come back up. However, when a write server is inactive for an extended period, then brought up, and query servers are synchronized, any changes made earlier on query servers are lost.

When login management is disabled, IQ continues to propagate settings to other servers, but does not perform checks like password expiration at login. GRANT and REVOKE propagate from the write server to other servers. GRANT CONNECT to change a user password also propagates from a query server to other servers. GRANT CONNECT to add a new user or REVOKE connect to drop a user will not propagate from a query server to other servers.

The database option MPX\_LOCAL\_SPEC\_PRIV enables you to add a user to a query server, but Sybase IQ does not propagate such users. If the same user is added to the write server, the current user on the query server is replaced by the one on the write server.

If you drop a user from a write server and the user owns objects on a query server, the user's password is set to null, and connection is denied. If a drop operation fails on more than one query server, that user is also locked out.

## Collisions in login management

When two actions with conflicting results execute at about the same on two servers in a multiplex, a collision occurs. To resolve collisions, one action takes effect and overwrites the other. Propagation of settings among the servers in the multiplex will result in the settings that match on all servers.

For example, suppose that a DBA on one server sets the default connection limit for new users to 10 and another DBA sets it to 15. Once propagation completes, the result is that either all servers see the setting as 10, or else all servers see it as 15.

Ideally, the change that was done last should persist, but in this situation, it is not critical for any of the login management features. Because SQL Remote propagates the changes, one of the changes fails and the earlier update persists.

---

**Warning!** Due to a known issue with SQL Remote, Sybase recommends that you avoid performing batches of login management operations simultaneously on multiple servers without synchronizing afterwards.

When a row or column is updated on a remote server twice within a minute, at the same time that the same row/column is updated on the consolidated server, the remote server stores the first value specified instead of the second value. The second update is not propagated from the SQL Remote update of the second value on the consolidated server. If this state occurs, synchronizing query servers or changing the value on any server corrects it.

Sybase recommends that, whenever possible, you perform login management changes on the write server. If necessary, you can make changes on any query server, but avoid making conflicting changes on another server until all query servers have been synchronized by SQL Remote.

---

For information about static collisions involving permissions, see "Setting multiplex permissions" on page 555.

## 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.

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. In Sybase Central, the server for the utility database is known as the Utility Server.

You start the utility database by specifying utility\_db as the database name when connecting. (Do not specify “utility\_db” as the database file, as there is no database file associated with the utility database.) 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.

---

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?

These topics are discussed in this section.

## Defining the utility database password

To use the utility database you must specify the user ID DBA. The password for the utility database is held in a file named *util\_db.ini*, which is stored in the server executable directory. As this directory is on the server, you can control access to the file, and thereby control who has access to the password.

The *util\_db.ini* file has the following contents:

```
[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 12-5: 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_asiq -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_asiq -n testsrv -gu utility_db -iqmt 256
```

Assuming that the utility database password has been set during installation to ASIQ, 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=ASIQ;dbn=utility_db;eng=testsrv" -jConnect
```

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

This section describes how to grant permissions to users and create new users using DBISQL 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, as 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 described in “Managing IQ user accounts and connections” on page 556.

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 DBISQL 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. For details, see “Adaptive Server Enterprise system and catalog procedures” on page 13. The ASE stored procedures do not let you use the Sybase IQ Login Management facilities for limiting connections.

## Creating new users

A new user is added to a database by the DBA using the GRANT CONNECT statement. For example:

❖ **Adding a new user to a database, with user ID M\_Haneef and password welcome**

- 1 From DBISQL, connect to the database as a user with DBA authority.
- 2 Issue the SQL statement:

```
GRANT CONNECT TO M_Haneef  
IDENTIFIED BY welcome
```

Only the DBA has the authority to add new users to a database.

Initial permissions for new users

By default, new users are not assigned any permissions beyond connecting to the database and viewing the system tables. In order to access tables in the database they need to be assigned permissions.

The DBA can set the permissions granted automatically to new users by assigning permissions to the special PUBLIC user group, as discussed in “Special groups”.

Using a DBISQL command file to set up new users

You may want to put commands for setting up new users into a DBISQL command file. Command files help you standardize the way you perform processes you repeat over time. For details on using command files, see Chapter 2, “Using Interactive SQL (dbisql),” in the *Sybase IQ Utility Guide*.

Creating users in Sybase Central

❖ **Creating a user in Sybase Central**

- 1 Connect to the database.
- 2 Open the Users & Groups folder.

- 3 Double-click the New User icon or choose File > New User. The User Creation wizard leads you through the process.

For more information, see the Sybase Central Online Help for the IQ plug-in.

## Changing a password

### Changing a user's password

If you have DBA authority, you can change the password of any existing user with the following command:

```
GRANT CONNECT TO userid IDENTIFIED BY password
```

The same command can also be used to add a new user. For this reason, 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 system procedures `sp_addlogin` and `sp_adduser` to add users. 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.

### Implementing 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 “VERIFY\_PASSWORD\_FUNCTION option” in the *Sybase IQ Reference Manual*.

To set a minimum password length, see “MIN\_PASSWORD\_LENGTH option” in the *Sybase IQ Reference Manual*.

### Changing the DBA password

The user ID DBA identifies a user with full administration and resource creation rights. The default password for user ID DBA for all databases is SQL. You should change this password to prevent unauthorized access to your database. The following command changes the password for user ID DBA to new\_password:

```
GRANT CONNECT TO DBA  
IDENTIFIED BY new_password
```

Extra steps are required to change the DBA password for a multiplex database.

To change the DBA password, you must have DBA authority.

---

**Warning!** Never drop the DBA user for a multiplex database. Doing so makes the database unusable.

---

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.

## Granting DBA and resource authority

DBA and RESOURCE authority are granted in exactly the same manner as each other.

❖ **Granting resource permissions to a user ID**

- 1 Connect to the database as a user with DBA authority.
- 2 Type and execute the SQL statement:

```
GRANT RESOURCE TO userid
```

For DBA authority, the appropriate SQL statement is:

```
GRANT DBA TO userid
```

### Notes

- Only the DBA can grant DBA or RESOURCE 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.
- You should give users with DBA authority two user IDs, one with DBA authority and one without, so that they connect as DBA only when necessary.
- RESOURCE authority allows the user to create new database objects, such as tables, views, indexes, or procedures.

## Granting 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.

### Example

All table and view permissions are granted in a very similar fashion. You can grant permission to M\_Haneef to delete rows from the table named sample\_table as follows:

- 1 Connect to the database as a user with DBA authority, or as the owner of sample\_table.
- 2 Type and execute the SQL statement:

```
GRANT DELETE  
ON sample_table  
TO M_Haneef
```

You can grant permission to M\_Haneef to update the column\_1 and column\_2 columns only in the table named sample\_table as follows:

- 1 Connect to the database as a user with DBA authority, or as the owner of sample\_table.
- 2 Type and execute the SQL statement:

```
GRANT UPDATE (column_1, column_2)  
ON sample_table  
TO M_Haneef
```

Table 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). 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.

**Granting user permissions on tables in Sybase Central**

One way to grant a user permissions on a table in Sybase Central is as follows:

**❖ Granting user permission on tables in Sybase Central**

- 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.

Legend for the columns on Permissions tab:

- A=Alter
- D=Delete
- I=Insert
- R=Reference
- S=Select
- U=Update

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.

## Granting users the right to grant permissions

Each of the table and view permissions described in “Granting permissions on tables and views” can be assigned WITH GRANT OPTION. This option gives the right to pass on the permission to other users. This feature is discussed in the context of groups in “Permissions of groups”.

### Example

You can 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, as follows:

- 1 Connect to the database as a user with DBA 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
```

## Granting permissions on procedures

There is only one permission that may be granted on a procedure, and that is the EXECUTE permission to execute (or CALL) the procedure.

Permission to execute stored procedures may be granted by the DBA or by the owner of the procedure (the user ID that created the procedure).

The method for granting permissions to execute a procedure is similar to that for granting permissions on tables and views, discussed in “Granting permissions on tables and views”.

### Example

You can grant M\_Haneef permission to execute a procedure named my\_procedure, as follows:

- 1 Connect to the database as a user with DBA 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.

Granting user permissions on procedures in Sybase Central

One way to grant a user permissions on a table in Sybase Central is as follows:

- ❖ **Granting user permissions on procedures in Sybase Central**
  - 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.

## Revoking user permissions

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. To disallow M\_Haneef from executing my\_procedure, the command is:

```
REVOKE EXECUTE ON my_procedure FROM M_Haneef
```

This command must be issued by the DBA or by the owner of the procedure.

Permission to delete rows from sample\_table can be revoked by issuing the command:

```
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 12 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'
```

- Never drop the DBA user for a multiplex database. Doing so makes the database unusable.
- 

## Managing groups

Once you understand how to manage permissions for individual users (as described in the previous section) working with groups is straightforward. A group is identified by a user ID, just like a single user, but this user ID is granted the permission to have **members**.

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, using the commands described in “Managing individual user IDs and permissions”.

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.

The ability to create a group without a password enables you to prevent anybody from signing on using the group user ID. This security feature is discussed in “Groups without passwords”.

## Creating groups

### ❖ **Creating a group with a name and password**

- 1 Connect to the database as a user with DBA authority.
- 2 Create the group's user ID just as you would any other user ID, using the following SQL statement:

```
GRANT CONNECT  
TO 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.

Creating groups in  
Sybase Central

### ❖ **Creating a group in Sybase Central**

- 1 Connect to the database.
- 2 Click the Users & Groups folder for that database.
- 3 Double-click Add Group. A Wizard leads you through the process.

For more information, see the Sybase Central online Help.

## Granting group membership to users

Making a user a member of a group is done with the GRANT statement.

Membership in a group can be granted either by the DBA or by the group user ID. You can grant user M\_Haneef membership in a group personnel as follows:

- 1 Connect to the database as a user with DBA authority, or as the group user ID personnel.
- 2 Grant membership in the group to M\_Haneef with the following SQL statement:

```
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.

Adding users to groups in Sybase Central

### ❖ Adding a user to a group in Sybase Central

- 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.

## 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.

Notes

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.

## Referring to 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.

### Creating a group to own the tables

It is advisable 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. For an example of this, see the section “Database object names and prefixes”.

## Groups without passwords

Users connected to a group's user ID have certain permissions. This user ID can grant and revoke membership in the group. Also, this user would 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 by the DBA, 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, the GRANT CONNECT statement is typed without a password. Thus:

```
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.

## Special groups

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 Chapter 9, “System Tables,” and Chapter 11, “System Views,” in *Sybase IQ Reference Manual*.

### 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.

## Database object names and prefixes

The name of every database object is an identifier. The rules for valid identifiers are described in the chapter “SQL Language Elements” in *Sybase IQ Reference Manual*.

In queries and sample SQL statements throughout this guide, database objects from the sample database are generally referred to using their simple name. For example:

```
SELECT *
FROM employee
```

Tables, procedures, and views all have an owner. The owner of the tables in the sample 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".employee
```

The employee 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. This section describes when you need to use the owner prefix to identify tables, view and procedures, and when you do not.

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
```

**Note**

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.

## Using views and procedures for extra security

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.

In these cases, you can use views and stored procedures to tailor permissions to suit the needs of your organization. This section describes some of the uses of views and procedures for permission management.

For information on how to create views, see “Working with views”.

## Using views for tailored security

Views are computed tables that contain a selection of rows and columns from base tables. 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 employee 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 the following:

```
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 which only looks at sales employees as follows:

```
CREATE VIEW emp_sales AS
SELECT emp_id, emp_fname, emp_lname
FROM "DBA".employee
WHERE dept_id = 200
```

The table should be identified as “DBA”.employee, with the owner of the table explicitly identified, for the SalesManager user ID to be able to use the view. Otherwise, when SalesManager uses the view, the SELECT statement refers to a table that user ID does not recognize.

- 3 Give SalesManager permission to look at the view:

```
GRANT SELECT
ON emp_sales
TO SalesManager
```

Exactly the same command is used 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 order_date, region, sales_rep, company_name
FROM "DBA".sales_order
```

```
KEY JOIN "DBA".customer
```

- 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 "DBA".emp_sales ;  
SELECT * FROM "DBA".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".employee ;  
SELECT * FROM "DBA".sales_order ;
```

#### Other permissions on views

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 “Using views” on page 251.

## Using procedures for tailored security

While views restrict access on the basis of data, procedures restrict the actions a user may take. As described in “Granting permissions on procedures” a user may have EXECUTE permission on a procedure without having any permissions on the table or tables on which the procedure acts.

#### Strict security

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. With this approach, the manner in which data in the database can be modified is strictly defined.

## Using procedures to disable connections

You can disable connections using the stored procedure `sp_iqmodifylogin`. For an example, see the *Sybase IQ Reference Manual*.

## 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?

Sybase IQ decides whether a user ID has permission to carry out a specific action in the following manner:

- 1 If the user ID has DBA permissions, the user ID can carry out any action in the database.
- 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.

This approach minimizes problems associated with the order in which permissions are set.

## Managing the resources connections use

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**.

**Setting options**

You can set database options using the SET OPTION statement, which has the following syntax:

```
SET [ TEMPORARY ] OPTION
... [ userid. | PUBLIC. ]option-name = [ option-value ]
```

For reference information about options, see Chapter 2, “Database Options,” in *Sybase IQ Reference Manual*. For information on the SET OPTION statement, see Chapter 6, “SQL Statements,” in *Sybase IQ Reference Manual*.

**Resources that can be managed**

The following database options can be used to manage resources. See Chapter 5, “Managing System Resources” in *Sybase IQ Performance and Tuning Guide* or see the *Sybase IQ Reference Manual* for more information on these options.

- **CURSOR\_WINDOW\_ROWS** Defines the number of cursor rows to buffer.
- **LOAD\_MEMORY\_MB** Sets an upper bound for the amount of heap memory that subsequent load operations can use.
- **MAIN\_CACHE\_MEMORY\_MB** Sets the size of the cache for the main IQ Store. (The server option -iqmc is the recommended way to set the main cache size.)
- **MAX\_CARTESIAN\_RESULT** Limits the number of result rows from a query containing a cartesian join.
- **MAX\_IQ\_THREADS\_PER\_CONNECTION** Sets the number of processing threads available to a connection for use in IQ operations.
- **TEMP\_CACHE\_MEMORY\_MB** 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.)
- **QUERY\_TEMP\_SPACE\_LIMIT** Limits the amount of temporary dbspace available to any one query.
- **QUERY\_ROWS\_RETURNED\_LIMIT** 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:

- **JAVA\_HEAP\_SIZE** Sets the maximum size (in bytes) of that part of the memory that is allocated to Java applications on a per connection basis.

- **MAX\_CURSOR\_COUNT** Limits the number of cursors for a connection.
- **MAX\_STATEMENT\_COUNT** Limits the number of prepared statements for a connection.

Database option settings are not inherited through the group structure.

## Other settings that affect user resources

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 “Managing IQ user accounts and connections” on page 556.

For more information on settings that affect memory and other resources available to user connections, see Chapter 5, “Managing System Resources” in *Sybase IQ Performance and Tuning Guide*.

## Users and permissions in the system tables

Information about the current users of a database and about their permissions is stored in the database system tables and system views.

For a description of each of these tables, see Chapter 9, “System Tables” in *Sybase IQ Reference Manual*.

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.

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
SYSDUMMY	PUBLIC	Dummy table, can be used to find the current user ID
SYSPROCPERM	PUBLIC	Each row holds one user granted permission to use one procedure

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
SYSGROUPS	PUBLIC	Information from SYSGROUP in a more readable format
SYSTABAUTH	PUBLIC	Information from SYSTABLEPERM in a more readable format

<b>View</b>	<b>Default</b>	<b>Contents</b>
SYSCOLAUTH	PUBLIC	Information from SYSCOLPERM in a more readable format
SYSPROCAUTH	PUBLIC	Information from SYSPROCPERM in a more readable format

The following table lists system tables used by Sybase IQ Login Management. The user DBA owns these tables, so their qualified names are DBA.IQ\_SYSTEM\_LOGIN\_INFO\_TABLE and so on.

<b>Table</b>	<b>Default</b>	<b>Contents</b>
IQ_SYSTEM_LOGIN_INFO_TABLE	DBA only	Each row holds Sybase IQ Login Management database defaults and status for connections, passwords, and locked logins.
IQ_USER_LOGIN_INFO_TABLE	DBA only	Each row holds Sybase IQ Login Management definitions and locked login status for one user
SYSOPTIONDEFAULTS	DBA only	Each row holds Sybase IQ default option settings.

In addition to these, there are tables and views containing information about each object in the database.

# Transport-Layer Security

## About this chapter

This chapter shows you how to secure communications between a client and the IQ server or between an IQ client and the database server using transport-layer security (TLS).

For more information about database security, see *SQL Anywhere Studio Security Guide* on the Sybase Product Manuals web site Product Manuals at <http://www.sybase.com/support/manuals/>.

## Contents

Topic	Page
About transport-layer security	590
Setting up transport-layer security	595
Invoking transport-layer security	596
Certificate authorities	600
Certificate chains	601
Enterprise root certificates	602
Globally signed certificates	605
Getting server-authentication certificates	606
Verifying certificate fields	608
Using transport-layer security for web services	610

## About transport-layer security

Transport-layer security, an IETF standard protocol, secures client/server applications using digital certificates and public-key cryptography.

Clients use trusted public certificates to encrypt data and authenticate servers in the initial client/server handshake. Data transmitted by the client can only be decrypted by the matching private key, which is stored in the database server certificate.

For server authentication, the database server sends its public certificate to the client. The client verifies the identity of the server using certificate fields and the digital signature embedded in the certificate.

**Efficiency** The transport-layer security standard overcomes the inefficiencies associated with public-key cryptography. Once a secure connection is established, the client and server exchange a common key. They use a highly efficient symmetric cipher for the rest of their communication.

**Supported platforms** To use transport-layer security, both server and client must be operating on Solaris, Linux, NetWare, or any supported 32-bit Windows platform except Windows CE, and the connection must be over the TCP/IP port.

FIPS-certified security options are available on Windows only.

## FIPS 140-2 certification

Federal Information Processing Standard (FIPS) 140-2 specifies requirements for security algorithms. It does not, however, specify requirements for security protocols such as SSL or transport-layer security. FIPS 140-2 is granted by the American and Canadian governments through the National Institute of Standards and Testing (NIST) and the Canadian Communication Security Establishment (CSE). Certicom has earned FIPS certification for security algorithms implemented on Windows.

Sybase IQ offers transport-layer security with the option of using the underlying FIPS-certified algorithms in the Certicom software.

You can use FIPS-certified security algorithms to encrypt your database files, or to encrypt communications for database client/server communication.

## About public-key cryptography

Public key cryptography makes use of mathematical systems that work with pairs of very large, associated numbers. These numbers, called **keys**, have particular properties. Each key can be used to encrypt information. Once encrypted, these messages can only be decrypted using the matching key.

One of the keys, called the **public key**, is published in a public forum. It can be used to encrypt information to be sent to the owner of the public key. The owner keeps the second key, called the **private key**, secret. A message encrypted with the public key can be decrypted only using the matching private key. Since the public key is published, anyone can create a message that only the owner of the private key can read.

In addition, a message encrypted with the private key can be decrypted by anyone who knows the public key. Such a message can be created only by someone who knows the private key. If the private key is kept secret, the owner can prove his or her identity by constructing such a message.

It is essential that the private key cannot be found easily through knowledge of the public key. The ease with which the private key can be derived from the public key is often associated with the strength of the cryptosystem and the size (in bits) of the public key. Another aspect of the private key is that it must be difficult to guess. The generation of high-quality private keys must incorporate pseudo-random data of high quality. If the data is predictable, it is easier for an adversary to guess the keys. To meet this criterion, the tools provided with Sybase IQ gather pseudo-random data from the operating system when generating new private and public key pairs.

### The role of public-key cryptography

Public-key cryptography has many advantages. Using the public key, anyone can send a message that can be read only by the person who knows the matching private key. Likewise, someone can prove that they know a private key by using it to encrypt a message. To verify the identity of a key owner, you can send an arbitrary message and ask them to encrypt. You can be sure that person knows the private key if you can decrypt the resulting message with their public key.

These features make public-key cryptography especially useful when establishing a secure communication link and happen automatically when you establish a synchronization connection using transport-layer security.

Once the secure link is established, the server and client automatically switch to a symmetric-key system of equivalent strength. In a symmetric system, the same key is used to encrypt and decrypt messages. This type of symmetric cipher can be computed more efficiently, reducing the computation time required to encrypt and decrypt messages.

#### How transport-layer security works

Transport-layer security works by filtering all incoming and all out-going communication through the cipher of your choice. The translation occurs between the Sybase IQ server and the communication protocol of your choice. For example, adding security to a TCP/IP connection affects the architecture as shown in the following diagram:

Transport-level security requires additional communication between a Sybase IQ client and the IQ server *before* the upload stream is sent. When a client initiates synchronization, it passes a message to the server. The client encrypts this message using the server's public key. The server decrypts this message using its private key. Initially, the server encrypts all messages to the client using the client's public key.

While this public-key/private-key cipher is secure as long as the private keys are kept secret, the encryption and decryption process is computationally intensive. To make further communication more efficient, the client and server agree upon and exchange another key and switch to a symmetric key cipher. They use this key and cipher for the rest of their communication because the symmetric cipher allows data to be encrypted and decrypted more efficiently.

## Client architecture

To synchronize with a Sybase IQ server by secure means, the client must use the same cipher suite as the server. All messages received from the server via a communication protocol, such as TCP/IP, are decrypted before being passed to the remote Sybase IQ client.

## Digital certificates

A **digital certificate** is an electronic document that identifies a person or entity and contains a copy of their public key. Each certificate includes a public key so that anyone can communicate securely with the person or entity by encrypting information with this public key. Digital certificates conform to a standardized file format that contains the following information:

- Identity information, such as the name and address of the certificate owner.
- Public key.
- Expiry date.
- One or more digital signatures.

Digital signatures	<p>A <b>digital signature</b> provides a means to detect whether a certificate has been altered. A digital signature is a cryptographic operation created by calculating a value, called a <b>message digest</b>, from the identity information and the public key.</p> <p>A message digest is a bit-value designed to change if any part of the certificate changes. The algorithm used to calculate the message digest is known to all users of the certificates. The correct value is encrypted with the private key contained in the certificate. Thus, anyone can detect alteration using the algorithm to calculate the message digest, using the public key to decrypt the message digest contained in the certificate, and comparing the two values.</p> <p>If you are using FIPS-approved RSA encryption, you must generate your certificates using RSA.</p>
The importance of digital certificates	<p>A certificate constructed in this manner is called a <b>self-signed certificate</b> because the digital signature is constructed with the matching private key. Such a certificate cannot be altered without knowledge of the private key.</p> <p>Digital certificates play the role of identity cards. The signatures prevent alteration because as long as the private keys used to create the signatures are kept secret, the digital certificate cannot be altered.</p>

## The role of digital certificates

A Sybase IQ server must be able to identify itself to clients with its own server certificate. The client must ensure that the certificate is authentic. To do so, the client must already have a trusted copy of the public certificate. Alternatively, the server's certificate may be signed by another certificate. In the latter case, the client must have a reliable copy of the signing certificate.

The Sybase IQ server must have access to its public certificate and to the private key for this certificate. This information is contained in a **server identity**. A server certificate is constructed by appending the private key to the matching public certificate.

The following figure displays a sample server certificate. This certificate is a server identity, suitable for use by a Sybase IQ server. This particular certificate has been signed by another certificate. The file contains both public certificates and the server's password.

Since other users may have access to the computer running the Sybase IQ server, the file containing the private key is protected by a password. This password is intended to maintain the honesty of the people given access to the computer. It does not provide an adequate barrier to an outside attack as the password is only a few characters in length. To further protect the private key, outsiders must be denied access to the Sybase IQ server by a firewall, or by other traditional means.

Instead of being signed directly by the certificate authority, the server's certificate may be the first certificate in a certificate chain. In this case, the client must trust the owners of all certificates and must have a trusted copy of the final certificate in the chain, called the root certificate. Such a certificate file would have a structure similar to that displayed above, but could contain a longer list of certificates.

## Using chains of certificates

A certificate may be signed by other certificates, or it may be **self-signed**, which means it is signed only with its own private key. A sequence of public certificates, each signed by the next, is called a **certificate chain**. At one end of a typical chain is a certificate used for a particular Sybase IQ server. At the other end is a certificate, signed by no other certificates, called the **root certificate**.

You can arrange certificates in various ways, depending on your requirements. The following sections describe how to construct and use certificate chains to achieve particular security goals. The following topics are covered:

- If you have only a single server, the simplest setup is to create a self-signed certificate. The only disadvantage is that the private key for the certificate must be held on the Sybase IQ server, where it is harder to protect.
- An enterprise root certificate is of particular benefit to organizations using more than one Sybase IQ server. In this setup, IQ clients need keep only a copy of this root certificate to recognize any server issuing a certificate signed by this root certificate.
- Commercial certificate authorities can benefit organizations that require the utmost in security. These organizations can help in two ways. First, the root certificates they use are of the highest possible quality, making these certificates somewhat less prone to attack. Secondly, commercial certificate authorities can provide a trusted third party when two companies wish to communicate securely but are not familiar with each other.

- You can, and in some cases should, use the facilities provided to verify certificate fields. This precaution is appropriate in many scenarios, but is particularly so when using a globally signed certificate. In this case, you are unlikely to want your clients to trust certificates that your certificate authority has signed for other customers.

In all cases, you must ensure that the Sybase IQ command line and log file are secure. This is best done using a firewall and by otherwise limiting access to the computer running the Sybase IQ server.

Sybase IQ transport-layer security is a flexible mechanism that lets you achieve the security important to your setup. The basic system allows you to keep information private, while certificates ensure IQ clients that they are talking to a trusted Sybase IQ server.

## Server authentication

One method of breaking a system is to masquerade as the server. The client connects to what it thinks is the server, but the connection is unknowingly made to another, hostile server. To guard against this form of attack, the server can use a digital certificate. A digital certificate plays the role of an identity card.

Each digital certificate contains a public encryption key and information about the owner's identity. The certificates are designed in such a way that they can be altered only by someone who knows the matching private key. As long as this private key is kept a secret, clients can safely assume the identity information accurately identifies a server. To ensure that they are talking to the correct server, clients ask the server to prove that it knows the matching private key. The server can do so by decrypting a message that has been encrypted with the public key shown in the certificate.

## Setting up transport-layer security

To set up Adaptive Server Anywhere transport-layer security, perform the following steps:

- **Create digital certificates** Create public certificates and server certificates. Public certificates are distributed to client applications, while server certificates are stored securely with database servers.

See “Digital certificates” on page 592.

- **Start the Adaptive Server Anywhere database server with transport-layer security** Use the -ec database server option to specify the type of security, the server certificate, and the password to protect the private key.  
See “Invoking transport-layer security” on page 596.
- **Configure client applications to use transport-layer security** Specify the path and file name of trusted public certificates using the Encryption connection parameter [ENC].  
See “EncryptedPassword connection parameter [ENP]” on page 146.

## Invoking transport-layer security

You can use transport-layer security when using the TCP/IP, HTTP, or HTTPS communication protocols. For TCP/IP, HTTP and HTTPS you must use RSA encryption.

### ❖ Using transport-layer security

To invoke transport-layer security, you must first set it up for the client, storing the settings in the publication, subscription, or Sybase IQ user.

- 1 Create digital certificates to invoke server authentication features.

Use utilities supplied with Sybase IQ to create public certificates that are distributed to client applications and server certificates that are stored securely with database servers. Different types of certificates and different arrangements of these certificates allow you to provide various levels of security.

- 2 Start the Sybase IQ server with transport-layer security.

Use the -ec startup switch on the start\_asiq command line to specify the type of security, the server certificate, and the password to protect the private key. Sybase IQ transport-layer security is available only over TCP/IP and on Solaris, Linux, or any supported Windows platform except Windows CE. For syntax, see Chapter 1, “Running the Database Server” in the *Sybase IQ Utility Guide*.

- 3 Configure client applications to use transport-layer security.

Specify the path and file name of trusted public certificates using the Encryption connection parameter [ENC]. For details, see “Encryption connection parameter [ENC]” on page 147.

The Certicom security software built into Sybase IQ uses certificates for the purpose of server identification. Sample certificates are provided with Sybase IQ for RSA encryption. The sample RSA certificate is called rsaserver.crt and the password is test.

---

**Warning!** The sample certificates should be used for testing purposes only. The sample certificates provide no security in deployed situations because they and their corresponding passwords are widely distributed with Sybase software. To protect your system, you must create your own certificate.

---

Confirming proper startup

If Certicom security starts properly, the informational messages confirm this fact. The absence of such messages indicates that Certicom security has not started properly.

## Self-signed certificates

Sybase IQ includes tools for working with certificates. A utility named gencert lets you generate new certificates. Since certificates are normally written in a machine-readable format, another utility, named readcert, displays the contents of a certificate in human-readable format.

You can make a number of types of certificates with the gencert utility. The easiest type to make is a self-signed (root) certificate, as no other signing certificate is required.

The main advantage of a setup with only one root certificate is simplicity; you need create only one certificate. This setup is often sufficient for simple setups involving only one Sybase IQ server. If you operate multiple IQ servers, an enterprise level certificate, discussed later, is often more convenient.

The biggest disadvantage is that a self-signed certificate is easier than other types to forge. This type of attack can be accomplished by creating a counterfeit certificate using a different key pair. Other types of certificates are more secure because they bear more than one digital signature.

## Making a new self-signed certificate

To generate a root certificate, start the gencert utility from a command prompt using the `-r` option. The utility prompts you to enter the identity information, the certificate password and expiry date, and the names of the new certificate files.

In the following procedure, you are prompted for names for the certificate, private key, and server identity files. Sybase IQ accepts any name and extension for these files. However, Windows only recognizes `.crt` and `.cer` extensions as certificate files.

In the following procedure, an RSA certificate is generated.

```
>gencert -r Certificate Generation Tool Choose  
certificate type ((R)SA or (E)CC): R Enter key length  
(512-2048): 2048 Generating key pair... Country: CA  
State/Province: Ontario Locality: Waterloo  
Organization: Sybase, Inc. Organizational Unit: IAS  
Common Name: MobiLink Serial Number: 2003.07.29.01  
Certificate valid for how many years: 2 Enter password  
to protect private key: password Enter file path to save  
certificate: self.crt Enter file path to save private  
key: self.pri Enter file path to save server identity:  
serv1.crt
```

The response to each question should be a string, except for the number of years to the expiry date, which must be an integer.

The utility creates three files, which in this example are called `self.crt`, `self.pri`, and `serv1.crt`.

- **self.crt** This file contains the new certificate, including the identity information, public key, expiry date, and signature. You can give out copies of this file to people whom you wish to contact you.
- **self.pri** This file contains the private key that matches the public key encoded in the certificate. The private key is encoded using the password you supplied, providing a modest barrier to others with access to your computer. However, since password encryption is not very secure, you must restrict access to this file to maintain secrecy.
- **serv1.crt** This file contains the same information as the above two files, combined into one file. It is intended for use with a Sybase IQ server. The server sends the public information to identify itself to clients. It requires the private key to decode messages returned by the clients. You must restrict access to this file. It, too, contains a copy of the private key, protected only by the password.

The server certificate contains the information in the public and private certificate files. You can make a server certificate by concatenating a public certificate and the file containing the private key.

## Using a self-signed certificate

You can use the self-signed certificate for server authentication by following these steps:

1 Supply a copy of the public certificate to all clients. When the client first contacts the Sybase IQ server, the server will send the client a copy of the public certificate, *self.crt*. The client can detect fake certificates by comparing the one sent by the server with the copy the client already has.

2 Tell each client that it is to trust only servers that can decrypt messages encoded using the public key contained within the copy of the supplied public certificate. For Sybase IQ clients, you do so using the *trusted\_certificates* security parameter. For example, you can tell an Sybase IQ client to trust only the *self.crt* certificate by including a parameter in the address clause of the synchronization subscription.

To tell a client to trust only the desired certificate, name the trusted certificate using the *-r* option at the command line when running the generator.

3 When you start the Sybase IQ server, specify the name of the server certificate file, *serv1.crt*, and the corresponding password. Open a command prompt and run the following command line:

```
start_asiq -c "dsn=Adaptive Server IQ  
Demo;uid=DBA;pwd=SQL" -x tcpip ( security=rsa_tls (   
certificate=serv1.crt; certificate_password=password )  
)
```

You can hide the contents of the command line using the File Hiding utility, dbfhid. For more information, see the *Adaptive Server Anywhere Database Administration Guide*.

Note that the clients do not need and should not have either the private key or the password that unlocks it. Clients need only the public certificate.

In contrast, the Sybase IQ server requires access to the private key, as well as to the public parts of the certificate. Thus, the server requires access to the server certificate file, which contains both public and private information.

The Sybase IQ server must have access to the private key and the password that protects it. For this reason, you must ensure that the Sybase IQ command line and log file are secure. This is best done using a firewall and by otherwise limiting access to the computer running the Sybase IQ server.

---

**Note** The certificate file name and password are not displayed in the log file.

---

## Certificate authorities

One problem with self-signed certificates is that an adversary can create a fake certificate using a different public- and private-key pair. Someone, mistaking the fake certificate for the original, may unknowingly encrypt his or her message using the substitute public key, rather than that owned by the intended recipient. Only the adversary, who knows the substitute private key, could read a message encrypted using the fake certificate.

To guard against such an attack, both the user and the owner of the certificate must agree to trust a third party. This third party, called a **signing authority** or **certificate authority**, adds a digital signature to the certificate using his or her private key. Once signed, the document certificate can be altered only with the aid of the third party. To sign a certificate, the certificate authority need not know the private key of the certificate owner.

The certificate authority need not be an external person or organization. If the certificates are to be used only within the company, it may be appropriate for someone at the company to act as the certificate authority.

To create a trustworthy system, a certificate authority must confirm the identity of a certificate owner before signing a certificate. In particular, the certificate authority must check that the identity fields in the certificate accurately describe the certificate owner and that the certificate owner owns the matching private key.

Someone wishing to use this certificate to communicate with the certificate owner must have confidence in the following:

- Before signing the certificate, the certificate authority made certain that the identity information contained in the certificate correctly identified the certificate owner.
- Each private key is known only to the certificate owner.

- The user has a reliable copy of the certificate authority's public key.

To satisfy these conditions, not only must the user have confidence in the integrity of the certificate authority, but the user must also have obtained the same public key directly from the certificate authority.

To obtain valid copies of a public key, users of this system typically obtain copies of a self-signed certificate owned by the certificate authority. To foil impostors, the certificate must be obtained by reliable means.

In addition, each client must store the copy of the certificate authority's certificate securely. Should an adversary have access to the user's computer, he or she could replace the certificate authority's certificate with a fake.

## Certificate chains

When deploying a replication system, a large number of certificates may be required. The responsibility of signing many certificates may place too great a burden on the certificate authority. To lessen their workload, a certificate authority can delegate signing authority to others. To do so, the certificate authority signs a certificate held by the delegate. The delegate then proceeds to sign certificates using the private key that matches the one in this certificate.

A certificate chain is a sequence of certificates such that each certificate is signed by the next. The final certificate, called the root certificate, is owned by a certificate authority. For example, a server certificate can be signed by a delegate. The delegate's certificate can be signed by a certificate authority. The certificate authority's public key is contained in a third certificate. Such a situation is a chain of three certificates.

In fact, a delegate can also have delegates. Thus, a chain of certificates can be of any length. However, the final certificate is always a self-signed root certificate, owned by a certificate authority.

To trust a chain, a user must trust each of the following:

- Before signing each certificate, the certificate authority and all delegates made certain that the identity information contained in the certificate correctly identified the certificate owner.
- Each private key is known only to the certificate owner.
- The user has a reliable copy of the certificate authority's public key.

All conditions are extremely important. The chain of certificates is only as strong as its weakest link.

## **Enterprise root certificates**

A deployment of Sybase IQ that involves multiple servers can be improved by assigning each server a unique certificate also signed by a common root certificate. A certificate authority within the enterprise holds the root certificate.

This arrangement has the following advantages:

- Each Sybase IQ server can be given a unique certificate, so that if one site is compromised, the others are not affected.
- Security is enhanced because the private key for the enterprise root certificate need not be stored on the Sybase IQ server.
- Clients do not need to keep a copy of each server's public certificate, only a copy of the public root certificate because you can configure them to trust any certificate signed by the root certificate.

The security of the system can be improved somewhat by obtaining a globally signed certificate, discussed later, from a commercial certificate authority. In practice, however, this arrangement provides adequate security for many applications.

You can program your clients to verify the values of some certificate fields, as discussed later. In this way, you can ensure that your clients synchronize with particular Sybase IQ servers within your organization.

This setup provides more flexibility than self-signed server certificates. For example, you can add a new server and give it a new certificate. If the new certificate is signed with the same enterprise root certificate, existing clients will automatically trust it. Were you, instead, to give each Sybase IQ server a self-signed certificate, all clients would require a copy of the new public certificate.

## Creating the certificates

The first step in setting up an enterprise-level system is to generate the common self-signed certificate. To generate this root certificate, start gencert with the `-r` option.

```
>gencert -r Certificate Generation Tool Choose  
certificate type ((R)SA or (E)CC): E Generating key  
pair... Country: CA State/Province: Ontario Locality:  
Waterloo Organization: Sybase, Inc. Organizational  
Unit: IAS Common Name: MobiLink Serial Number:  
2003.07.29.02 Certificate valid for how many years: 2  
Enter password to protect private key: password2 Enter  
file path to save certificate: ent_root.crt Enter file  
path to save private key: ent_root.pri Enter file path  
to save server identity: ent_serv.crt
```

The utility creates three files, which in this example are called `ent_root.crt`, `ent_root.pri`, and `ent_serv.crt`.

- **ent\_root.crt** This file contains the new certificate. This certificate should be published as all clients require a reliable copy.
- **ent\_root.pri** This file contains the private key that matches the public key encoded in the certificate.
- **ent\_serv.crt** This file contains the same information as the above two files, combined. It is intended for use with a Sybase IQ server.

You can use the first two of the three files to sign additional, new certificates. To generate a signed certificate, start gencert with the `-s` option. Type the name of the signing certificate file, the name of the signing private-key file, and the password for the signing private key.

```
>gencert -s Certificate Generation Tool Choose  
certificate type ((R)SA or (E)CC): E Generating key  
pair... Country: CA State/Province: Ontario Locality:  
Waterloo Organization: Sybase, Inc. Organizational  
Unit: IAS Common Name: MobiLink Serial Number:  
2003.07.29.03 Certificate valid for how many years: 1  
Enter file path of signer's certificate: ent_root.crt  
Enter file path of signer's private key: ent_root.pri  
Enter password for signer's private key: password2 Enter  
password to protect private key: password3 Enter file  
path to save server identity: serv1.crt
```

This time, gencert creates only one file. This file contains the signed certificate and the private key. It is intended for use with a Sybase IQ server.

Repeat this last step as many times as necessary to create a signed certificate for each Sybase IQ server.

```
>gencert -s Certificate Generation Tool Choose  
certificate type ((R)SA or (E)CC): E Generating key  
pair... Country: CA State/Province: Ontario Locality:  
Waterloo Organization: Sybase, Inc. Organizational  
Unit: IAS Common Name: MobiLink Serial Number:  
2003.07.29.04 Certificate valid for how many years: 2  
Enter file path of signer's certificate: ent_root.crt  
Enter file path of signer's private key: ent_root.pri  
Enter password for signer's private key: password2 Enter  
password to protect private key: password4 Enter file  
path to save server identity: serv2.crt
```

You now have the following files:

- **ent\_root.crt** The root certificate.
- **ent\_root.pri** The root private key.
- **ent\_serv.crt** The root combined certificate.
- **serv1.crt** The combined certificate for the first Sybase IQ server.
- **serv2.crt** The combined certificate for the second Sybase IQ server.

You do not need the combined root certificate because no Sybase IQ server uses it directly. Instead, you created a separate certificate for each Sybase IQ server.

## Using the signed certificates

You can use the signed certificates for server-authentication by following these steps:

- 1 Supply a copy of the public root certificate to all clients. When the client first contacts the Sybase IQ server, the server sends the client a copy of its own public certificate. This certificate bears the signature of the root certificate. The client can detect fake certificates by verifying that the root signature matches the public key in their copy of the root certificate.
- 2 Tell each client that it is to trust only servers whose certificates bear the signature of the root certificate. For Sybase IQ clients, use the `trusted_certificates` security parameter. For example, you can tell a Sybase IQ client to trust only the `ent_cert.crt` certificate by including a parameter in the address clause of the synchronization subscription.

To tell a client to trust only the desired certificate, name the trusted certificate using the `-x` option at the command line when running the generator.

- 3 When you start each Sybase IQ server, specify the name of that server's certificate file and the corresponding password. Enter each command on one line.

```
start_asiq -c "dsn=Adaptive Server IQ
Demo;uid=DBA;pwd=SQL" -x tcpip ( port=3333;
security=rsa_tls ( certificate=serv1.crt;
certificate_password=password3 ) ) start_asiq -c
"dsn=Adaptive Server IQ Demo;uid=DBA;pwd=SQL" -x tcpip
( port=4444; security=rsa_tls (
certificate=serv2.crt; certificate_password=password4
) )
```

## Globally signed certificates

You can improve the security of a multi-server Sybase IQ setup by assigning each server a unique certificate that is signed by a common root certificate. You can improve it further using a certificate signed by a commercial certificate authority. Such a certificate is called a global certificate or a globally-signed certificate. A commercial certificate authority is an organization that is in the business of creating high-quality certificates and using these certificates to sign other certificates.

A global certificate has the following advantages:

- Security requires that both parties trust the root certificate. In the case of inter-company communication, common trust in an outside, recognized authority may increase confidence in the security of the system because a certificate authority must guarantee the accuracy of the identification information in any certificate that it signs.
- Security is enhanced when keys are created using pseudo-random data of high quality. The data used with the gencert utility is of cryptographic quality, but other, even better methods can be used in controlled environments.
- The private key for the root certificate must remain private. An enterprise may not have a suitable place to store this crucial information, whereas a certificate authority can afford to design and maintain dedicated facilities.

When using a globally signed certificate, each client must verify certificate field values to avoid trusting certificates that the same certificate authority has signed for other clients. This process is described in the next section.

## Getting server-authentication certificates

Sybase IQ transport-layer security is based on Certicom SSL/TLS Plus libraries, which require elliptic-curve or RSA certificates. You can get a global certificate from any certificate authority that can supply certificates in the correct format. Two such companies are VeriSign and Entrust Technologies.

For more information, see <http://www.verisign.com/> at <http://www.verisign.com/> or [http://www.entrust.com/certificate\\_services/index.htm](http://www.entrust.com/certificate_services/index.htm) at [http://www.entrust.com/certificate\\_services/index.htm](http://www.entrust.com/certificate_services/index.htm).

There are several ways to get certificates. One way is to use the Certicom reqtool utility, which is installed when you install the security component. This tool creates a server certificate and a global certificate request. Copy the contents of the public certificate onto your clipboard, and paste them into the form on the Web site of the certificate-issuing authority. Only submit the public component of the certificate request. You must not disclose your private key.

For more information about this procedure, see the document *reqtool.pdf*, located in the *win32* subdirectory of your Sybase IQ installation. It is installed when you install the security component.

### Example

The following example creates an elliptic-curve certificate:

```
> reqtool -- Certicom Corp. Certificate Request Tool  
3.0d1 -- Choose certificate request type: E - Personal  
email certificate request. S - Server certificate  
request. Q - Quit. Please enter your request [Q] : S  
Choose key type: R - RSA key pair. D - DSA key pair.  
E - ECC key pair. Q - Quit. Please enter your request  
[Q] : E Using curve ec163a02. Generating key pair  
(please wait)... Country: CA State: Ontario Locality:  
Waterloo Organization: Sybase, Inc. Organizational  
Unit: IAS Common Name: MobiLink Enter password to  
protect private key : password5 Enter file path to save  
request : global.req Enter file path to save private key  
: global.pri
```

The file *global.req* contains the public certificate and request information. Paste the contents of this file into the form on the certificate-issuing Web site.

The file *global.pri* contains the private key for the enterprise certificate. This file is protected by the password you entered, but since the protection provided by the password is weak, you must store this file in a secure location.

## Using a global certificate as a server certificate

You can use your global certificate directly as a Sybase IQ server certificate. To do so, you must create a server identity certificate by concatenating the public and private certificates. Open a command prompt and run the following command line:

```
copy global.crt+global.pri global2.crt
```

You can now start a Sybase IQ server, specifying the new certificate and the password for your private certificate. Open a command prompt and run the following command line:

```
start_asig -c "dsn=Adaptive Server IQ  
Demo;uid=DBA;pwd=SQL" -x tcpip ( security=rsa_tls(  
certificate=global2.crt;  
certificate_password=password5 ) )
```

You can hide the contents of the command line using the File Hiding utility, dbfhid. For more information, see the *Adaptive Server Anywhere Database Administration Guide*.

You must also ensure that clients contacting your Sybase IQ server trust the certificate. To do so, you must tell the clients to trust the root certificate in the chain. In this case, the root certificate in the chain is a certificate held by the certificate authority.

By default, clients trust certificates signed by the Sybase root certificate used to sign the sample certificate included with Sybase IQ.

For better security, however, you should ensure that clients consider only the root certificate of your certificate authority to be valid.

To tell a client to trust only a particular root certificate, name the trusted certificate using the *-r* option at the command line when running the generator.

## Verifying certificate fields

Global certificates have one potentially serious flaw. Because the Sybase IQ clients, as configured above, trust all certificates signed by the certificate authority, they may also trust certificates that the same certificate authority has issued to other companies. Without a means to discriminate, your clients might mistake a competitor's Sybase IQ server for your own and accidentally send it sensitive information.

Similar precautions can be required in other scenarios. A company may use an enterprise certificate, but it may still be important to verify with which department a Sybase IQ client is connected.

This problem can be resolved by requiring your clients to test the value of fields in the identity portion of the certificate. Three fields in the certificate can be verified. You can verify any or all of the following three fields:

- Organization
- Organizational Unit
- Common Name

To verify the fields, you supply the acceptable value.

## Verifying fields in certificate chains

When the first certificate is part of a chain, all the specified field values are checked in that certificate. If specified, the company name is also checked in all the other certificates, except for the root certificate. This arrangement allows for the case that the root certificate is held by a certificate authority. In this case, the field values of the root certificate will be different, as it is owned by the certificate authority, rather than your company or organization.

## Using a globally-signed certificate as an enterprise certificate

Instead of using a global certificate as a server certificate, it is possible to instead use it to sign other certificates, as you would an enterprise certificate. This setup lets you combine the benefits of a global certificate and an enterprise certificate. The most important advantage is that you need not store the private key for your global certificate on the computer running the Sybase IQ server.

To create such a setup, generate a unique certificate for each Sybase IQ server. When you do so, sign them with your global certificate.

The following example displays how two server certificates can be generated and signed by the global certificate:

```
>gencert -s Certificate Generation Tool Choose  
certificate type ((R)SA or (E)CC) : E Generating key  
pair... Country: CA State/Province: Ontario Locality:  
Waterloo Organization: Sybase Organizational Unit: IAS  
Common Name: MobiLink Serial Number: 2003.07.29.06  
Certificate valid for how many years: 1 Enter file path  
of signer's certificate: global.crt Enter file path of  
signer's private key: global.pri Enter password for  
signer's private key: password5 Enter password to  
protect private key: password6 Enter file path to save  
server identity: serv6.crt >gencert -s Certificate  
Generation Tool Choose certificate type ((R)SA or  
(E)CC) : E Generating key pair... Country: CA  
State/Province: Ontario Locality: Waterloo  
Organization: Sybase Organizational Unit: IAS Common  
Name: MobiLink Serial Number: 2003.07.29.07 Certificate  
valid for how many years: 1 Enter file path of signer's  
certificate: global.crt Enter file path of signer's  
private key: global.pri Enter password for signer's  
private key: password5 Enter password to protect private  
key: password7 Enter file path to save server identity:  
serv7.crt
```

The above commands generate two server identity certificates, intended for use with two Sybase IQ servers.

- **serv6.crt** The server identity certificate for Sybase IQ server #1.
- **serv7.crt** The server identity certificate for Sybase IQ server #2.

Both certificates are signed by *global.crt*, which in turn is signed by your certificate authority's root certificate.

You can start these two Sybase IQ servers with the following commands, entered one command per line.

```
start_asiq -c "dsn=Adaptive Server IQ  
Demo;uid=DBA;pwd=SQL" -x tcpip (  
port=3333;security=rsa_tls ( certificate=serv6.crt;  
certificate_password=password6 ) ) start_asiq -c  
"dsn=Adaptive Server IQ Demo;uid=DBA;pwd=SQL" -x tcpip  
( port=4444;security=rsa_tls ( certificate=serv7.crt;  
certificate_password=password7 ) )
```

You can hide the contents of the command line using the File Hiding utility, dbfhide. For more information, see the *Adaptive Server Anywhere Database Administration Guide*.

In addition, you must ensure that each client trusts your certificate authority's root certificate.

## Using transport-layer security for web services

### ❖ Setting up transport-layer security for web services

- 1 Create digital certificates.

Create both public and server certificates. Public certificates (which can be Certificate Authority certificates) are distributed to browsers or web clients. Server certificates are stored securely with your Adaptive Server Anywhere web server. See “Creating the certificates” on page 603.

- 2 Start the web server with transport-layer security.

Use the -xs database server option to specify HTTPS, the server certificate, and the password to protect the private key. For syntax, see “Starting the database server” in the *Sybase IQ Utility Guide*.

- **Start the web server with transport-layer security**

Following is a partial start\_asiq command line.

```
-xs protocol(Certificate=server-
certificate;Certificate_Password=password;...) ...
```

- **protocol** can be https, or https\_fips for FIPS-approved RSA encryption. https\_fips uses a separate approved library, but is compatible with https.

---

#### Note

The Mozilla Firefox browser can connect when https\_fips is used. However, the cipher suite used by https\_fips is *not* supported by the Internet Explorer, Opera, or Safari browsers—if you are using https\_fips, these browsers cannot connect.

---

- **server-certificate** The path and file name of the server certificate. For HTTPS, you must use an RSA certificate.

- **password** The password for the server certificate's private key. You specify this password when you create the server certificate.
- **Configure web clients** Configure browsers or other web clients to trust public certificates. The trusted certificate can be self-signed, an enterprise root, or a Certificate Authority certificate.

For general information about creating digital certificates, including information about using Certificate Authorities, see “Digital certificates” on page 592.

#### Example

The following steps show how to use a certificate with HTTPS synchronization.

- 1 Obtain an RSA server certificate file. For example, obtain a file called *server\_cert.crt* with password *pwd*.
- 2 Obtain a public RSA certificate file. For example, obtain a file called *client\_cert.crt*.
- 3 On the *start\_asiq* command line, include the following:

```
-x  
https(certificate=server_cert.crt;certificate_password=pwd)
```

Instead of exposing the password in the command line, you can use the *dbfhide* utility. For more information, see “File Hiding utility (dbfhide)” in the *Sybase IQ Utility Guide*.

- 4 In the synchronization user or the synchronization subscription, use the following type and address:

```
... TYPE https ADDRESS  
"trusted_certificates=client_cert.crt"
```



# Data Backup, Recovery, and Archiving

## About this chapter

This chapter explains how to back up and recover a database and how to use read-only hardware to archive non-modifiable data for easy access. It tells why it is important to perform backups on a regular basis and gives recommendations for scheduling backups.

## Contents

Topic	Page
Backup protects your data	613
Virtual Backups	632
Performing system-level backups	634
Validating your database	637
Restoring your databases	639
Unattended backup	652
Getting information about backups and restores	653
Determining your data backup and recovery strategy	656
Backing up and restoring the multiplex	660
Multiplex server migration and failover	663
Archiving data with read-only hardware	670

## Backup protects your data

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. Recovery from system failures is discussed in *Sybase IQ Troubleshooting and Recovery Guide*.

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.

## **Backing up your database**

You use the BACKUP command to back up your IQ database. Backup includes both the Sybase IQ data (the IQ Store) and the underlying Adaptive Server 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 in order to back it up. The BACKUP command has no way to specify another database.

For an IQ multiplex database, you must run backups of the write server on the write server, but you may execute backups while the servers are all running in multiplex mode.

## **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 Adaptive Server Anywhere database objects you add. It is known as the Catalog Store, and has the dbspace-name SYSTEM.

- *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.
- *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.
- Multiplex users may also create an IQ Local Store on a query server. It behaves like the IQ Store with respect to only that query server, and shares buffer cache with the IQ Main Store. This optional store has no default extension, although many users choose *.iqloc*.

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)

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**

This procedure summarizes backup steps. Read the rest of this chapter for complete details before you perform a backup.

- 1 Connect to the server using an account with DBA privileges. For a multiplex database, you must connect to the write server.
- 2 Run the BACKUP command. For complete syntax, see the *Sybase IQ Reference Manual*.

It backs up the following files:

- Catalog Store (SYSTEM dbspace file), typically named *dbname.db*
  - The transaction log file, which is required for system recovery, typically named *dbname.log*
  - 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 write server
    - IQ Temporary Store

❖ **Backing up and restoring query servers with IQ Local Stores**

Enter commands in the following procedure without line breaks.

- 1 Connect to the query server using dbisql or dbisqlc. Enter the following command:

```
BACKUP DATABASE FULL to 'full_path_to_backup_file'
```

- 2 Stop the query server.
- 3 In the write server's directory, enter the following from a Command Prompt:

```
sync_qnode SQL query_server_name
```

- 4 In the query server's directory, enter the following command:

```
dbbackup -y -d <full_path_of_the_query_server_dir> -  
c  
"uid=DBA;pwd=SQL;eng=<writer_name>;links=tcpip{host  
=<writer_host>;port=<writer_port>}"
```

- 5 Delete the log file. For example:

```
rm <database>.log
```

If the log file or database name on the query server is different from the write server's files, use the dblog utility to set the transaction log file name. You may use the relative path of the query server's database file. For more about dblog, see Chapter 3 in the *Sybase IQ Utility Guide*. For example:

```
dblog -r -t all_types.log all_types.db
```

- 6 Start the query server without the database as:

```
start_asiq @params.cfg -n query_server_name -x  
'tcpip{port=query_server_port}'
```

- 7 Connect to the query server using dbislc or dbisql and specify the utility database in the connect string. Enter the following command:

```
RESTORE DATABASE '<database_name>' FROM  
'<full_path_to_backup_file>'
```

- 8 Stop the query server that was started without the database.
- 9 Restart the query server via Sybase Central.

## Data in backups

*BACKUP backs up committed data only.* Backups begin with 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. However, if a failure occurs between the first and second checkpoints, any work that occurred after the first checkpoint cannot be restored.

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** Adaptive Server Anywhere databases use the transaction log and other logs differently. If you are recovering such a database, you need its transaction log file, and BACKUP retains it for you. See the *Adaptive Server Anywhere User's Guide* 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.

---

## Live backup of transaction log

You can make a live backup of the transaction log using the dbbackup utility with the -l option.

### ❖ Making a live backup of a transaction log

- 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 distributed evenly 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 “Restoring in the correct order” on page 647 for more information.

---

## Ensuring 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 “Validating your database” on page 637 for details.

## Selecting archive devices

You can back up any IQ database onto magnetic tape or disk, including WORM devices. (For more about WORM devices, see “Archiving data with read-only hardware” on page 670.) 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.

## 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.

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 *Sybase IQ 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
```

## 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 as described in “Granting DBA and resource authority”.

If you are backing up the IQ Store and Catalog Store on a multiplex database, you need to log onto the write server using an account with DBA privileges.

## Rewinding 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.

## Retaining 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. See “Unattended backup” for details if you want to run backups when no operator is present.

---

**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. For details, see Chapter 18, “Automating Tasks Using Schedules and Events.”

---

## Estimating 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 Adaptive Server 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. See “Unattended backup” for details.

For an operator-attended backup to tape, BACKUP simply begins the backup. If it runs out of room, you must mount additional tapes.

## **Concurrency and backups**

Backups can be run concurrently with most other database operations. Exceptions are:

- No metadata changes can occur while the Catalog Store is backed up
- No commands that issue checkpoints or DBCC

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.

## The BACKUP statement

To back up an IQ database, use the following syntax:

```
BACKUP DATABASE
...[ CRC ON | OFF ]
...[ ATTENDED ON | OFF ]
...[ BLOCK FACTOR integer ]
...[ { FULL | INCREMENTAL | INCREMENTAL SINCE FULL } ]
... [ { VIRTUAL DECOUPLED |
      VIRTUAL ENCAPSULATED 'shell_command' } ]
...TO 'archive_device' [ SIZE #_of_KB ][STACKER #_of_drives_in_stack
] ...
[ WITH COMMENT 'string' ]
```

---

**Note** If you are recovering an Adaptive Server Anywhere database, see “Backup and Data Recovery” in *Adaptive Server Anywhere User’s Guide* for additional options.

---

### Specifying 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. See the information on these options under “Specifying archive devices”.

### Specifying 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 ENCAPSULATED options of the BACKUP statement. For instructions, see “Virtual Backups” on page 632.

- 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 virtual backup is supported using the VIRTUAL DECOUPLED and VIRTUAL ENCAPSULATED options of the BACKUP statement.

- 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.

For guidance in selecting a backup type, see “Scheduling routine backups”.

## Specifying virtual backup

The VIRTUAL DECOUPLED | VIRTUAL ENCAPSULATED '*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. For more information on virtual backups, see “Virtual Backups” on page 632.

## Specifying archive devices

The TO *archive\_device* clause indicates the destination disk file(s) or system tape drive(s) for the backup. You specify one TO *archive\_device* clause for each destination file or device. At least one is required. BACKUP distributes output *in parallel*—that is, concurrently—across all of the devices you specify. For faster backups you should specify more devices, up to the number your hardware platform supports. While BACKUP attempts to distribute information equally across multiple devices, there is no guarantee that this will happen.

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 pathnames to specify the location of disk files. BACKUP interprets the pathname 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.

---

For more information about fixed-length I/O on Windows, see “Tuning backup operations” in the *Sybase IQ Performance and Tuning Guide*.

#### Specifying 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.

	<p>On Windows, there are special requirements for the SIZE option on tape devices:</p> <ul style="list-style-type: none"><li>• The value of SIZE must be a multiple of 64. Other values are rounded down to a multiple of 64.</li><li>• If you do not specify SIZE explicitly, it is automatically set to 1.5GB.</li></ul>
Specifying the size of disk backups	<p>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 <i>archive_device</i> is larger than the default of 2GB (UNIX) or 1.5GB (Windows).</p> <p>During backup, when the amount of information written to a given <i>archive_device</i> 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.</p> <p>For example, if you specify one <i>archive_device</i>, a disk file called <i>janfull</i>, and you specify SIZE 200000 for a maximum 200MB file, but your backup requires 2GB, then BACKUP creates ten 200MB files: <i>janfull.1</i>, <i>janfull.2</i>...<i>janfull.10</i>. You must ensure that your disk can accommodate this much data before performing the backup.</p>
Specifying stacker devices	<p>The STACKER option of the TO clause indicates that you are backing up to an automatically loaded multtape 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.</p>
Specifying devices for third party backups	<p><b>Note</b> Do not specify SIZE or STACKER if you are using a third party backup product, as size information is conveyed in the <i>vendor_specific_information</i> string. To specify this string, see “Performing backups with non-Sybase products.”.</p>

## Other backup options

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.

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.
Adding 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, <i>backup.syb</i> , that Sybase IQ provides.
	If you need to back up an Adaptive Server Anywhere-only database, see the <i>Adaptive Server Anywhere Reference Guide</i> for additional BACKUP options.

## Waiting 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 will be wholly contained by a previous depends-on backup. Read-only dbspaces that have been altered since the time of the depends-on backup will be backed up.

For more information, see “Archiving data with read-only hardware” on page 670.

## Backup examples

### Example 1 — Full backup

This example makes a full, attended backup of the database `asiquuser` 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 `asiquuser`. Then issue the following command:

```
BACKUP DATABASE
TO '/dev/rmt/0n'
TO '/dev/rmt/1n'
WITH COMMENT 'Jan 18 full backup of asiquuser'
```

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 asiquuser'
```

An example of how to restore this database from these two backups is provided later in this chapter.

## **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 as follows:

- 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. See “Recording dbspace names” for details on how to record this information after you complete a backup.

## 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 *Sybase IQ 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 at <http://www.sybase.com/support/techdocs/>.

---

## **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.

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.)

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.

## **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.

## 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. The 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.

### ❖ Performing an Encapsulated Virtual 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

- 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.

## 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.

### ❖ Performing a Decoupled Virtual Backup

- 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**

- 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.

## **Performing 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 procedures in this section, however, 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.

## Shutting down the database (non-multiplex)

Your IQ database must be shut down during a system-level backup.

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.

## Shutting down the database (multiplex)

Your multiplex write server must not be running during a system-level backup. To use Sybase Central during the system-level backup, simply connect it to one of the query servers.

You must shut down your write server before starting the system-level backup. You must also ensure that no one starts the write server until the system-level backup is complete. For details about shutting down write servers, see “How to stop the server” on page 62.

## Backing up the right files

Required files

You must back up the following files:

- All SYSTEM dbspace files, typically named *dbname.db*.
- 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 IQ\_SYSTEM\_MAIN

Save the lengths of the following files:

- The IQ\_SYSTEM\_TEMP dbspace file, typically named *dbname.iqttmp*

- Files for any additional dbspaces that have been added to IQ\_SYSTEM\_TEMP

It is not required that you back up the temporary dbspaces. 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

It is a good idea to back up the ASCII message files such as *dbname.iqmsg* and the *\$ASLOGDIR/\*.srvlog* and *\$ASLOGDIR/\*.stderr* files. These files are not required. However, if problems occur during a restore, the *.iqmsg* file 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 (IQMSG\_LOG\_SIZE option is greater than 0), 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.

Multiplex backup list

You will need to back up the IQ Local dbspaces on each query server. (The last step of both database level and system level backups is to synchronize query servers.) You should also record the sizes of files used for IQ temporary dbspaces at all of the query servers. Multiplex message files you may wish to preserve include the message log and dbremote log files stored in each server directory (same directory as the one holding the *dbname.iqmsg* file.)

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.

## 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. See “Shutting down the database (non-multiplex)” on page 635 for details. When restoring a multiplex database, you must shut down all the query 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. See the discussion of required and optional files in “Backing up the right files” on page 635.

In the case of the temporary dbspace files, ensure that files or raw devices are present with the correct filenames (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.

### Multiplex synchronization

After the system-level restore of a multiplex write server, start the write server and perform a multiplex synchronization to restore and start the query servers. You must then restore any IQ Local dbspaces on query servers. For details about synchronization, see “Synchronizing query servers” on page 204.

## 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 three different verification modes that perform increasing amounts of consistency checking. 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 following table summarizes the actions, output, and speed of the three sp\_iqcheckdb verification modes.

Mode	Errors detected	Output	Speed
allocation	allocation errors	allocation statistics only	4TB per hour
check	allocation errors most index errors	all available statistics	60GB per hour
verify	allocation errors all index errors	all available statistics	15GB per hour

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. For details on using sp\_iqcheckdb and interpreting the sp\_iqcheckdb output, see Chapter 2, "System Recovery and Database Repair," in the *Sybase IQ Troubleshooting and Recovery Guide*. For the complete syntax of sp\_iqcheckdb, see Chapter 10, "System Procedures" in the *Sybase IQ Reference Manual*.

#### Validating a multiplex database

Run sp\_iqcheckdb only on the write server of an IQ multiplex. If you run sp\_iqcheckdb on a multiplex query 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. For information on using the resources parameter, see the section "Resource issues running sp\_iqcheckdb" in Chapter 2, "System Recovery and Database Repair" of the *Sybase IQ Troubleshooting and Recovery Guide*.

If other users are active when you run sp\_iqcheckdb, the results you see reflect only what your transaction sees.

## 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

Before you can restore a database, make sure that the following conditions are met:

- You must have DBA privileges.
- The server and database that you wish to restore must be shut down.
- You must be connected to the utility\_db database. For information on utility\_db and how to set privileges for using it, see the *Sybase IQ Installation and Configuration Guide* for your platform. For instructions on starting utility\_db, see “Creating a database with SQL” on page 183.
- 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 Catalog Store (by default the *.db* file) and the transaction log (by default the *.log* file) *must not* exist in the location you are restoring to. If either of these files exists, you must delete it or move it 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 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, you must follow the procedure described in “Moving database files” on page 643. 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 give 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. Restoring individual files is not supported. However, you can move database files to a new location, using the RENAME clause of the RESTORE command.

## Restore accommodates dbspace changes

During a set of incremental restores, RESTORE creates and drops tablespaces 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 filename 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 12.0 and up. Due to changes in the format of the database, Sybase IQ 12.0 does not let you restore from a backup created on an 11.x version. Likewise, you cannot restore an 11.x IQ database from a Version 12.0 backup.

To move your data from a Sybase IQ 11.5.1 database to Sybase IQ 12.7, you must:

- 1 Upgrade to 12.5 using the migration procedure described in the version 12.5 *Sybase IQ Installation and Configuration Guide*.
- 2 Follow the migration procedure for 12.5 databases described in the 12.7 *Sybase IQ Installation and Configuration Guide*.

RESTORE does not let you restore an Sybase IQ backup to an Adaptive Server Anywhere database.

## The RESTORE statement

To restore a database, use the following syntax:

```
RESTORE DATABASE 'db_file'  
FROM 'archive_device' [ FROM 'archive_device' ]...  
... [ RENAME dbspace_name TO 'new_dbpath' ]...  
... [ CATALOG ONLY ]
```

Remember that 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 pathname or a pathname relative to the directory where the database was created. If you specify a new pathname, 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.

---

See “Specifying archive devices” for details on specifying devices.

---

**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 *asiquser.db*. The database is restored from two disk files. All database files are restored to their original locations.

```
RESTORE DATABASE 'asiquser.db'  
FROM 'c:\\asiq\\backup1'  
FROM 'c:\\asiq\\backup2'
```

## Moving database files

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, you use the Transaction Log utility (dblog). For syntax and details, see “Transaction Log utility (dblog)” in *Sybase IQ Utility Guide*.
- 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\\\\asiqnew.db'
  FROM 'c:\\asiq\\\\backup1'
  FROM 'c:\\asiq\\\\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 pathname, 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 IQ\_SYSTEM\_MAIN dbspace, which holds the main IQ Store file, are restored relative to the main IQ Store file.
- If you do not specify a RENAME clause, and the file was created using an absolute pathname, 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 *dbspace\_name* as it appears in the SYSFILE table. You specify *new\_dbspace\_path* as the new raw partition, or the new full or relative pathname, for that dbspace.

You cannot use the RENAME option to specify a partial restore.

Relative pathnames in the RENAME clause work as they do when you create a database or dbspace: the main IQ Store dbspace, Temporary Store dbspaces, and Message Log are restored relative to the location of *db\_file* (the Catalog Store); user-created IQ Store dbspaces are restored relative to the directory that holds the main IQ dbspace.

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.

See “Recording dbspace names” for information on how to obtain a list of the dbspace names in your database, so that you know the correct names to include in RENAME clauses.

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 dbspace 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, as described in “Restoring in the correct order”.

The first RESTORE command is:

```
RESTORE DATABASE 'asiquser'  
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 'asiquser.iqmsg'  
RENAME IQ_USER TO '/dev/rdsk/c1t5d2s1'
```

The second RESTORE command, to restore the incremental backup, is:

```
RESTORE DATABASE 'asiquser'  
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 'asiquser.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.

---

## 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. See “Content of the backup log” on page 654 for a list of the information displayed.

When you specify CATALOG ONLY you must include the FROM *archive\_device* clause, but omit the RENAME clause.

## 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. See “Getting information about backups and restores” for the content and location of this file.

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.

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<sub>1</sub>, A<sub>2</sub>, and A<sub>3</sub>. Tapes A<sub>1</sub> and A<sub>2</sub> contain the Catalog Store. This set must be restored first, and must be in the first device.

**Set B** Tapes B<sub>1</sub> and B<sub>2</sub>. 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<sub>1</sub>, C<sub>2</sub>, and C<sub>3</sub>. 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.

## 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 DBISQLC or DBISQL 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_asiqdemo'
```

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 DBISQL (Java).

---

## 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 filename is changed.

---

You can also use dblog to rename the transaction log, even if you have not restored the database, given certain restrictions. For details, refer to the section “Transaction Log utility (dblog)” in the *Sybase IQ Utility Guide*.

You can access the Transaction Log utility from the system command line, using the dblog command-line utility. For full syntax of the dblog utility, see the section “Transaction Log utility (dblog)” in Chapter 3, “Database Administration Utilities” of the *Sybase IQ Utility Guide*.

## 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. For more information, see “Validating your database” on page 637.

## 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 “Backing up and restoring the multiplex” on page 660.

## 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.

For an example of the information you see in a header file, see any RESTORE line in the sample backup log in “Content of the backup log”. A RESTORE with CATALOG ONLY produces the information in the same format as the backup log entry for an actual RESTORE.

## 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.

## 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 asiqdemo.db
-rw-r--r--  1 fiona  sybase   122880000 Feb 26 18:27 asiqdemo.iq1
-rw-r--r--  1 fiona  sybase   122880000 Feb 26 18:27 asiqdemo.iq2
-rw-r--r--  1 fiona  sybase   122880000 Feb 26 18:27 asiqdemo.iq3
-rw-r--r--  1 fiona  sybase   122880000 Feb 26 18:27 asiqdemo.iqttmp
-rw-r--r--  1 fiona  sybase   122880000 Feb 26 18:27 asiqdemo.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 asiqdemo.iq1 ->
```

```
LINKS/asiqdemo.iq1
lrvwxrwxrwx 1 fiona sybase      14 Feb 26 17:48 asiqdemo.iq2 ->
LINKS/asiqdemo.iq2
lrvwxrwxrwx 1 fiona sybase      18 Feb 26 17:48 asiqdemo.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.

## Unattended backup

With the ATTENDED OFF option, you can specify that no operator will be present during a backup. Sybase IQ currently 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.

---

## 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 “Displaying header information”.

---

## 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 ASLOGDIR 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 ASLOGDIR 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 the following fields, separated by commas:

- Operation (Backup or Restore)
- Version
- Database name
- Database type (Sybase IQ or Adaptive Server 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 Adaptive Server 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, '2001-01-31 16:25:00.000', DBA, Full, Arch,
```

```
TED_FULL00, ''
BACKUP, 2.0, all_types.db, ASIQ, '2001-01-31 16:53:00.000', DBA, Incr, Arch,
TED_X_bkup_inc, ''
RESTORE, 2.0, all_types.db, ASIQ, '2001-01-31 16:25:00.000', DBA, Full, Arch,
TED_FULL00, ''
RESTORE, 2.0, all_types.db, ASIQ, '2001-01-31 16:53:00.000', DBA, Incr, Arch,
TED_X_bkup_inc, ''
BACKUP, 2.0, all_types.db, ASIQ, '2001-01-31 20:07:00.000', DBA, InSF, Arch,
A_partial2_yes_sf, ''
BACKUP, 2.0, all_types.db, ASIQ, '2001-01-31 20:07:00.000', DBA, InSF, Arch,
A_partial2_yes_sf, ''
```

## 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 ASLOGDIR 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 tablespaces, you need to know the name of every tablespace in the database. The tablespace names are in the SYSFILE table of every database, but you do not have this table available when you are restoring. For this reason, you should execute the sp\_iqdbspace stored procedure or issue the following statement any time you back up your database:

```
select * from SYSFILE
```

Keep the results of this query some place other than the disk where the database resides, so that you have a complete list of tablespace names if you need them.

You can also run the following script in DBISQL. 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.

---

```
-- This select statement will get names of IQ dbspaces  
and file names  
-- and add rename syntax including quotes  
  
select 'rename' , dbspace_name , 'to' , ''' + file_name  
+ ''''  
from SYSFILE where store_type ='IQ';  
  
-- output to file in proper format  
-- no delimiters and no additional quotes  
  
output to restore.tst delimited by ' ' quote '';  
  
--this produces a file restore.tst looking like this:  
--rename IQ_SYSTEM_MAIN to '/dev/rdsk/c2t0d1s7'  
--rename IQ_SYSTEM_TEMP to '/dev/rdsk/c2t1d1s7'  
--rename IQ_SYSTEM_MSG to 'all_types.iqmsg'
```

## 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

## 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 to 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

BACKUP and RESTORE write your IQ data in parallel to or from all of the archive devices you specify. The Catalog Store is written serially to the first device. Faster backups and restores result from greater parallelism. To achieve greater performance when backing up or restoring a large database, specify more 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 Adaptive Server 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.

# Backing up and restoring the multiplex

This section describes:

- Notes on back up and restore for multiplex servers and databases.
- Special restrictions that apply to backup and restore operations in a multiplex environment

In addition to the usual reasons for backing up and restoring data, you can also use the restore operation to recreate the multiplex on a different system when no problems have occurred.

### Multiplex backup list

Back up the IQ Store as described in “Types of backups” on page 615. You should also back up the IQ Local Stores on each query server. The last step of both IQ-level and system-level restores is synchronizing the query servers to propagate changes.)

You may wish to preserve the *server.dbrlog.NNN* files (stored in the write server's directory under */repDirs/logfiles* on UNIX or *\repDirs\logfiles* on Windows).

Updating your backup list

It is critically important 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 later, 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 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.

After each database backup, you may wish to truncate the transaction log, as described in *Sybase IQ Performance and Tuning Guide*.

❖ **Restoring the multiplex**

To restore the multiplex, you must first restore the Catalog Store and IQ Store, then restore any IQ Local Store(s) on query servers.

- 1 Confirm with Technical Support that a restore operation is needed.  
If you have trouble opening your database on a query server, try doing a synchronize operation first. For details, see “Synchronizing query servers” on page 204.
- 2 Confirm that database home directories for each server still exist. If not, create them or restore them from file system backups:
- 3 Shut down every server in the multiplex (write server and query servers) using the Stop Multiplex command in Sybase Central.

---

**Note** If automatic startup is enabled in your ODBC configuration, users on the same machine as the server may be able to start the server automatically and you will need to prevent this from happening while you are restoring the database.

---

- 4 After stopping all servers, verify that the database shut down successfully. If you see an active *asiqsr12* process with the server name of a server in the multiplex (in a startup parameter) you should stop it.

To verify on a UNIX system, use the *ps* command. For example:

```
% ps -ef | grep asiqsr12  
fiona 434      1  1  May 19 ?? 0:05 start_asiq -n myhost_myserver  
-c 32MB -x tcpip(port=1234) mpxdb.db  
fiona 4751    442  1 16:42:14 pts/5    0:00 grep start_asiq
```

To verify on a Windows system, use Task Manager. Look on the Processes tab for *asiqsr12.exe* or find the IQ Server icon in the system tray and stop it using right-click and Shutdown.

- 5 Move files required for debugging and reconfiguring the multiplex.
  - Make a file system copy of the *.iqmsg* file.
  - On each server, preserve any *dbname.iqtmp* dbspace files, to reconfigure the multiplex. If the IQ Temporary store is damaged, use the start the server with the *-iqnotemp* switch to drop and recreate the temporary store dbspaces. For more information, see *Sybase IQ Release Bulletin*.

Delete the following files from the write server:

```
<database_home>/<dbname>.db  
<database_home>/<dbname>.log
```

If a query server is damaged, however, drop it and re-add it after RESTORE but do not synchronize it.

- 6 Start the utility database from the write server directory using the write server's server name:

```
% start_asiq -n thoreau_Server01 -c 32MB  
-x tcpip(port=1234)
```

If you are restoring the database to a different home directory or on a different machine from the one that created the backup, you must start the utility database using the multiplex override switch (*-iqmpx\_ov*)

- 7 Connect to the utility database (*utility\_db*).

```
% dbisql -c  
"eng=thoreau_Server01;uid=DBA;pwd=SQL;dbn=utility_d  
b"
```

- 8 Run the RESTORE command. Moving the database (restoring to a different machine) requires the RENAME clause. For details, see the *Sybase IQ Reference Manual*.

- 9 Shut down the write server.

- 10 Make sure that the temporary dbspaces exist as before, on raw devices or as files of the correct length. See “Backing up the IQ Store and Catalog Store” on page 616. For information on starting without the IQ Temporary Store, see *Sybase IQ Release Bulletin*.
- 11 Start the write server and, *if restoring to the same location*, synchronize the multiplex. For more information, see “Synchronizing query servers” on page 204.
- 12 If restoring to a new location, you must connect to the write server using Interactive SQL (not Sybase Central) and drop the query servers after starting the multiplex. This removes the absolute paths to query servers in their former locations from the system tables. (If you do not know the names of the query servers, open the Multiplex container in Sybase Central to list them. Ignore any warnings that occur in this situation.)

To remove each query server, run the `sp_iqmpxdropqueryserver` stored procedure in `dbisql` or `dbisqlc` for each query server. For example, to drop a query server named `iqmpx_qs`, enter:

```
sp_iqmpxdropqueryserver ('iqmpx_qs')
```

If you wish to make the database multiplex, you may create new query server(s).

## Multiplex server migration and failover

The section describes how to plan for evolution of the multiplex and failover of multiplex components in case you lose one of the machines in the multiplex environment. It discusses

- Resources crucial to continued operation of the database, and how to protect them
- Procedures for migrating a multiplex server
- Procedures for failing over a multiplex server to a different machine

In a multiplex environment, disk resources for the write server are the most important. Failures that affect only resources local to a query server are easy to address since the query server can be easily dropped and then recreated with working resources.

## Strategy for loss of a machine

The worst hardware failure for a multiplex would be losing the write server's machine. This section suggests strategies for providing the necessary resources for the write server to continue operation after hardware failures. These alternatives offer trade-offs between cost and the length of the resulting service outage.

## Resource requirements

A write server requires the following disk resources:

- Raw device access to the shared disk devices that make up the IQ Store.
- File system access to the Catalog Store files (*.db* and *.log* files) that comprise an Adaptive Server Anywhere database used to hold all of the catalog information for IQ.
- Raw or file system access to the disk devices or files that make up the IQ Temporary Store.
- File system access to the SQL Remote files (repDirs directory under the write server home directory).

---

**Note** Multiplex databases built with Sybase Central under IQ version 12.4.3 and earlier may place the Catalog Store files in directories under a shared top-level directory. This is no longer needed.

---

To make recovery and failover easier, an IQ database administrator should keep careful notes about how all the disk and file system resources are set up in each server of the multiplex environment.

The IQ software is designed so that, as long as the four categories of disk resources hold contents that represent their state at a single point in time, the database server can restart and bring the database back to an operational state. In the case of the files or devices that make up the IQ Temporary Store, the server can even reconstitute the store if one or more devices fail, since the IQ Temporary Store holds no persistent data beyond what it can find in the Catalog Store.

The following subsections discuss several ways to use disk or tape resources to allow the return of the database to an operational condition.

## Stable disk storage

For the disk resources in the multiplex, it is best to use systems with built-in protection against hardware failures. Techniques for this protection include disk mirroring, RAID (redundant array of independent devices) disk arrays, and logging file systems.

The goal is to preserve disk contents in the most recent state, in terms of completed I/O write operations for the database server, despite any failures in components of the disk subsystem. With this guarantee, you can always restart the database, because the server will perform automatic crash recovery to bring the database to a consistent state.

Without stable disk storage, failures in the disk resources will require you to restore the disk resources to an earlier, known good state. The two ways to do this are OS-level backup and database backup. The following sections discuss these techniques.

## OS-level backup

An OS-level backup makes a copy, on alternate media, of the entire set of disk resources that the database was using *at a single point in time*. In general, you cannot make a consistent OS-level backup while any of the disk resources of the database is changing. In IQ multiplex, this means that you must stop the write server to keep it from modifying any of its disk resources during the OS-level backup. Query servers, however, may continue to run since they cannot modify the shared IQ Main disk resources, and their Catalog and Temporary Stores are private and need not be backed up. However, if the query server has an IQ Local Store, it must also be shut down.

## Database backup

The database BACKUP command produces a transactionally-consistent image of all the data in the database, and can run simultaneously with normal database operations. In a multiplex, the write server performs the database backup while query servers may continue to execute user queries. Because it must trace out the active data in the database, the database backup typically takes longer than an OS-level backup, though it may use less space on the backup media. The backup facility of Sybase IQ does allow most of the backup operation to run in parallel if it can use multiple backup devices, which can significantly speed up backups. For details about running BACKUP, see “Backing up and restoring the multiplex” on page 660.

## **Loss of the write server's machine**

In the event of a failure of the write server's machine, it is possible to recover to the point of the failure. This requires obtaining access to the write server's disk resources from a working machine.

If the write server can be repaired and brought back up with all the database resources operational, then it is only necessary to restart the write server and the dbremote processes.

If the write server cannot be repaired, you may still be able to reconstitute a working write server by reconnecting the failed machine's disk resources to a new machine. You can accomplish this by recabling disk drives, or by reconfiguring the logical layout of a shared disk-storage subsystem.

## **Migration**

Migration is the controlled movement of a server in a multiplex from one machine to another. All resources of the machines involved are assumed to be available during the entire procedure. There are two cases: migrating a query server, and migrating a write server.

### **Moving a query server to a different machine**

#### **❖ Moving the query server**

- 1 Open the multiplex folder and highlight the query server.
- 2 Right-click the server, then select File > Delete (or right-click on the server icon and select Delete). Click Finish in the pop-up window to delete the server. For details, see “Deleting query servers” on page 211.
- 3 Add the query server back to the multiplex on the new machine using the Create Query Server wizard. For details, see “Creating databases with multiplex functionality” on page 193.

### **Replacing write servers**

To replace the write server of a multiplex, you designate one of the query servers to become the new write server. While converting it, you must change the server name it uses. The new write server must have a server name that differs from any query servers already defined in the multiplex. You can use the former write server's name.

If the write server failed in such a way that its SYSTEM dbspace and transaction log files cannot be recovered, use the procedure “Replacing a write server that has no SYSTEM dbspace and/or transaction log file” on page 668. When the former write server's SYSTEM dbspace and transaction log files are available, use “Replacing a write server with intact files,” as follows:

❖ **Replacing a write server with intact files**

- 1 Stop all the servers in the multiplex using Sybase Central. For details, see “Managing multiplex servers” on page 207.
- 2 Use FTP or some other system facility to copy the write server's SYSTEM dbspace file(s) (*dbname.db*, typically, is the only one) to the new write server's directory.

You must copy the *dbname.db* to the query server directory that will be promoted. It cannot be copied to any other directory. If the writer's db/log has a name that is different from the query server's db/log, then the name must be changed when it is copied over.

- 3 Delete the transaction log file (*dbname.log*) in the new write server's directory and copy the transaction log file (*dbname.log*) from the old to the new write server.
- 4 Using the dblog facility, ensure that the log file path is set correctly for the new write server. If you use transaction log mirroring, also check the setting for mirroring at the new write server.
- 5 Start the query server that will become the write server in single-node mode (-iqmpx\_sn 1), as follows:

```
start_asiq @params.cfg -n <query-server> -iqmpx_sn 1  
-iqmpx_ov 1 -x 'tcpip{port=<q-port>}' <db-file-name>
```

- 6 Connect to the server as DBA and run the stored procedure:

```
sp_iqmpxreplacewriteserver('new-write-server').
```

- 7 Stop the server and restart it using the new write server name.

```
start_asiq @params.cfg -n <new-write-server>  
-iqmpx_sn 1 -iqmpx_ov 1 -x 'tcpip{port=<q-port>}'  
<db-file-name>
```

- 8 Connect as DBA and drop the old query server's dbspace definitions using:

```
sp_iqmpxdropserverdbspaces('<query-server>')
```

- 9 Stop the server.

- 10 Start Sybase Central, then start the write server and synchronize all the query servers. If no query servers remain, synchronization is not required.
- 11 If you use the administrative scripts that Sybase Central created, recreate them.

❖ **Replacing a write server that has no SYSTEM dbspace and/or transaction log file**

If a catastrophic loss of the write server occurs where the SYSTEM dbspace and transaction log files cannot be recovered, forced recovery is needed.

Replace the write server as follows:

- 1 Stop all the servers in the multiplex using Sybase Central. For details, see “Managing multiplex servers” on page 207.
- 2 Choose the query server that was most recently synchronized (the one with the largest value in “DBA”.*IQ\_MPX\_STATUS.catalog\_version*) to be the new write server. This preserves as much data as possible. Start this query server in single-node mode with the forced recovery and drop-leaks switches, as follows:

```
start_asiq @params.cfg -n <query-server> -x  
'tcpip{port=<w-port>}' -iqfrec <db-file-name>  
-iqdroplks <db-file-name> -iqmpx_sn 1 -iqmpx_ov 1  
<db-file-name>
```

- 3 Connect to the server as DBA and run the following stored procedure:

```
sp_iqmpxreplaceserver('new-write-server')
```

- 4 Recover free space in the IQ Store by running the command:

```
sp_iqcheckdb('allocation database resetclocks')
```

The execution time varies according to the size of the database, the number of tables or indexes specified, and the size of the machine.

- 5 Stop the server.
- 6 If you wish to enable log mirroring for the new write server or its log file, use the dblog utility.
- 7 Restart the server using the new write server name:

```
start_asiq @params.cfg -n <new-write-server> -x  
'tcpip{port=<w-port>}' -iqmpx_sn 1 -iqmpx_ov 1 <db-  
file-name>
```

- 8 Connect as DBA and drop the old query server's dbspace definitions using:

```
sp_iqmpxdropserverdbspaces('<query-server>')
```

- 9 Stop the server and start Sybase Central and synchronize all the query servers. If no query servers remain, synchronization is not required.
- 10 If you use the administrative scripts that Sybase Central created, recreate them.

## Recovery from loss of a multiplex server

Failover must occur when a server's machine has suffered a hardware failure that prevents it from continuing. As before, there are two cases: when the failed machine had a query server on it, and when it had the write server on it.

### Moving a query server from a failed to a good machine

Delete the failed query server from the IQ multiplex, and add a new one on a working machine, as in "Moving a query server to a different machine" on page 666. The failed query server need not be running.

### Write server failure where some disk resources are lost

"Recovery from loss of a multiplex server" describes the write server failover procedure when all the necessary disk resources can be made available on a new machine. If some of those resources are unavailable after a failure, and you choose not to use the replace write server procedure on page 668, use the standard database restore procedures in "Backing up and restoring the multiplex" on page 660. To diagnose database startup problems, verify the consistency of databases, and repair databases, see the *Sybase IQ Troubleshooting and Recovery Guide*.

You may protect your multiplex database using either database or OS-level backups. Certain disk-mirroring schemes available with shared-disk subsystems (such as the Timefinder or SRDF facilities from EMC) provide a very fast OS-level disk backup/restore service.

If restoring the database from IQ-level backups entails moving one or more dbspaces of the shared IQ store to new device(s), you may need to use the RENAME clause of the RESTORE command. You may then have to delete and re-add the query servers to let them access the new device(s).

## Archiving data with read-only hardware

Recent regulations such as the Health Insurance Portability and Accountability Act (HIPAA) specify rigid rules for data retention and compliance, requiring data archived in an 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.

## Dbspace access modes

Sybase IQ lets you alter dbspaces between read-write, read-only, and relocate modes. For details, see “Changing the mode of a dbspace” on page 232.

- Sybase IQ creates dbspaces in **read-write mode** by default.

- Sybase IQ always opens read-only dbspaces in **read-only mode**. The server startup switch `-iqro` causes the main or local store to be opened read-only, regardless of the status of the dbspaces in the main or local store. Sybase IQ opens read-only dbspaces with read-only access at the operating system level.

---

**Note** Sybase IQ 12.6 servers allowed dbspaces to be placed in read-only mode, but IQ did not enforce this at the volume or file system level. Sybase IQ 12.6 would not allocate from a read-only dbspace and would not write to a read-only dbspace. However, if the database store was opened in read-write mode, the read-only dbspace is actually open read-write at the file system level. This is the case for all stores open in read-write mode, except the main store of a multiplex reader or the main store of a database started with the `-r` (read-only) switch.

---

- Sybase IQ designates dbspaces in **relocate mode** as read-only in the sense that IQ will no longer allocate space from, or write to, the dbspace. It also designates the dbspace as a data source for the `sp_iqrelocate` stored procedure, which moves data from relocate dbspaces to read-write dbspaces.

Backup and restore operations support WORM devices. For details, see “Backup and restore using read-only hardware” on page 628.

## Effect of altering dbspaces on access modes

When you alter a dbspace from read-write to read-only mode, or vice versa, the operating system level access mode changes.

`ALTER DBSPACE READWRITE | READONLY` involves a transition period during which the dbspace is closed and reopened. By the time the `ALTER` statement commits, the dbspace is open in the new access mode. The exception is when the sole remaining read-write dbspace is altered read-only. In that case, some database structures must be written during the checkpoint command that follows the `ALTER` statement.

Recovery of `ALTER DBSPACE` replays the statement if the server fails before completing the checkpoint after the `ALTER`. In that case, the dbspace first opens in the “before” mode during database open. The recovery replay alters the dbspace to the “after” mode.

The read-write status of a dbspace is stored in the dbspace file header. Dbspaces in relocate mode open with read-write access at the operating system level. Since the file header contains the dbspace mode, Sybase IQ rewrites the file header to alter the dbspace from relocate to read-only mode. Keeping the dbspace open in read-write mode while it is in relocate mode facilitates this transition.

Sybase IQ allows the last dbspace of the main store to be altered read-only. All of the dbspaces of the main store may be made read-only. See “Using read-only hardware” on page 674.

In each store, one dbspace contains special structures owned by the database. For the IQ Main and IQ Local Stores, these structures are the freelist root page, the two database identity blocks, and the multiplex cm block (which exists in non-multiplex databases as well). In the IQ Temp Store, the special dbspace contains the temporary freelist root page.

The special structures are normally on a read-write dbspace. If the dbspace containing the special structures is altered read-only, the structures will move to another read-write dbspace. If the sole read-write dbspace is altered read-only, these structures remain on that dbspace while it is in read-only mode and while there are no other read-write dbspaces. The special structures will never be located on a dbspace in relocate mode.

## Transitions between access modes

The following restrictions apply to transitions between read-write, read-only and relocate modes:

- **Temporary dbspaces:**  
A database does not require a temporary dbspace. However, most commands require some temporary working space and return an error if no temporary dbspace exists. If a database does have one or more temporary dbspaces, there must be at least one read-write dbspace.
- **Altering dbspaces:**
  - The sole read-write main dbspace may be altered read-only. There must be no active, committed or applied read-write transactions in the transaction manager or an error results. The stored procedure `sp_iqtransaction` displays transaction status.
  - The sole read-write dbspace may not be altered relocate.
  - A relocate mode dbspace may be altered read-write or read-only.

- A read-only dbspace may be altered relocate unless it contains the special database structures.
- Alter read-write of any read-only dbspace that is a read-only hardware device that is “frozen” or of any device that does not have write privilege fails with an OS file open error.
- Any modification of any object in an IQ store fails if there are no read-write dbspaces in that store. Objects include table, index or internal database structures such as the backup identity object. Database backups require a read-write dbspace.
- Dropping dbspaces:
  - Tables and indexes may be dropped or modified even if they are wholly contained on read-only and/or relocate dbspaces.
  - Read-only and relocate mode dbspaces may be dropped if they are empty. If a read-only dbspace contains the special structures, it is not empty, and the structures will only move if a new dbspace is created in that store or if an existing dbspace of that store is altered read-write. The stored procedure `sp_iqdbspace` may be used to display the location of the special structures. Drop of a read-only dbspace does not require read-write access to the device.
- Verification and recovery:
  - `sp_iqcheckdb` does not require read-write space. `sp_iqcheckdb` will work normally when there are no read-write main dbspaces.
  - `-iqdropiks` will not update the freelist if there is no read-write main dbspace. The `sp_iqcheckdb` command, however, completes normally and reports results.
  - Forced recovery does not require read-write dbspace to open the database. However, if all main dbspaces were read-only both before and after the forced recovery, the database fails at the first checkpoint when it tries to write the rebuilt freelist.

Avoid altering a read-write dbspace to read-only mode if there are active read-write transactions. After completion of all read-write transactions that were active when the dbspace was read-write, you may alter it to relocate mode and then from relocate to read-only mode.

The server parameter `-iqro 1` opens the dbspaces of the writeable main store, i.e. the main store of a single-node or multiplex write server database or the local store of a multiplex query server database, in read-only mode. All read-write commands, including `ALTER DBSPACE`, require the writeable IQ Main Store be in read-write mode. The `-iqro` and `-r` options open the IQ Temporary Store in read-write mode, but the catalog dbspaces in read-only mode.

## 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

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.

### ❖ Creating 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

Suppose that you need to examine the archived database as of time  $t0$ .

- 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

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.

- 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

Create a new archive at time  $t1$  as follows.

- 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.

Perform the steps in “Creating an archive” followed by the steps in this section to create any number of archived versions of *db.db*.

# Sybase IQ as a Data Server

## About this chapter

Sybase IQ supports client application connections through either ODBC or JDBC. This chapter describes how to use Sybase IQ as a data server for client applications.

With certain limitations, Sybase IQ may also appear to certain client applications as an Open Server. This chapter also briefly describes the restrictions for creating and running these applications.

For information on developing Open Client applications for use with Sybase IQ, see “The Open Client Interface” in *Adaptive Server Anywhere Programming Interfaces Guide*.

The facilities described in this chapter do not provide remote data access for IQ users on Windows and Sun Solaris systems. Remote data access is provided by Component Integration Services (CIS), the core interoperability feature of OmniConnect(TM). For information on remote data access and proxy databases, see Chapter 16, “Accessing Remote Data” and Chapter 17, “Server Classes for Remote Data Access.”

## Contents

Topic	Page
Client/server interfaces to Sybase IQ	677
Setting up Sybase IQ as an Open Server	685
Characteristics of Open Client and jConnect connections	687

## Client/server interfaces to Sybase IQ

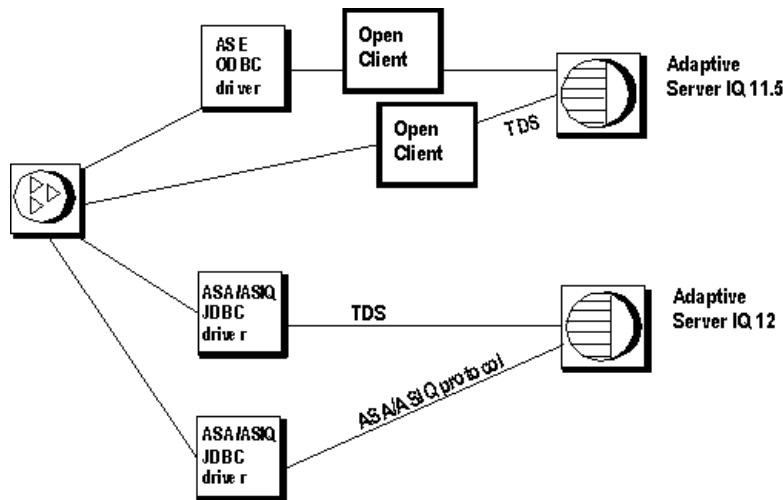
This section describes the key concepts of the Sybase IQ client/server architecture, and provides the conceptual background for the rest of the chapter.

If you simply wish to use a Sybase application or a third-party client application with Sybase IQ, you do not need to know any details of connectivity interfaces or network protocols. However, an understanding of how these pieces fit together may be helpful for configuring your database and setting up applications. This section explains how the pieces fit together, and avoids any discussion of the internal features of the pieces. For more details about third party client applications, see the *Sybase IQ Installation and Configuration Guide*.

#### Open Clients and Open Servers

Members of the Adaptive Server family act as **Open Servers**. Client applications communicate with Open Servers using the **Open Client** libraries available from Sybase. Open Client includes both the Client Library (CT-Library) and the older DB-Library interfaces. Sybase IQ can also act as an Open Server, but *in order to use the Open Client libraries, the client application must use only the supported system tables, views and stored procedures*. See Appendix A, “Compatibility with Other Sybase Databases,” in *Sybase IQ Reference Manual* for a list of compatible syntax.

The following figure shows how client applications communicate with Sybase IQ. In Sybase IQ 12, you can connect through either ODBC or JDBC. This contrasts with Sybase IQ 11.5 and earlier, which did not support JDBC.



#### Programming Interfaces and application protocols

Sybase IQ supports two application protocols:

- An application protocol specific to Sybase IQ and Adaptive Server Anywhere is used for ODBC, JDBC, and Embedded SQL applications.
- TDS (**tabular data stream**) is used for JDBC connections, Open Client applications and for other Sybase applications such as OmniConnect.

Tabular Data Stream	<p>Open Clients and Open Servers exchange information using the TDS application protocol. All applications built using the Sybase Open Client libraries are also TDS applications, because the Open Client libraries handle the TDS interface. However, some applications (such as Sybase jConnect) are TDS applications even though they do not use the Sybase Open Client libraries (they communicate directly to the TDS layer).</p> <p>At the other end of the client/server connection, while many Open Servers use the Sybase Open Server libraries to handle the interface to TDS, some applications have a direct interface to TDS of their own. Sybase Adaptive Server Enterprise and Sybase IQ both have internal TDS interfaces. They appear to client applications as an Open Server, but do not use the Sybase Open Server libraries.</p>
TDS uses TCP/IP	<p>Application protocols such as TDS sit on top of lower level communications protocols that handle network traffic. Sybase IQ supports TDS only over the TCP/IP network protocol. In contrast, the Sybase IQ-specific application protocol supports several network protocols as well as a shared memory protocol designed for same-machine communication.</p>

## Configuring IQ Servers with DSEdit

Sybase IQ can communicate with other Adaptive Servers, Open Server applications, and client software on the network. Clients can talk to one or more servers, and servers can communicate with other servers via remote procedure calls. In order for products to interact with one another, each needs to know where the others reside on the network. This network service information is stored in the interfaces file.

### The interfaces file

The interfaces file is usually named *sql.ini* on PC operating systems, and *interfaces* or *interfac* on UNIX operating systems.

The interfaces file is like an address book. It lists the name and address of every database server known to Open Client applications on your machine. When you use an Open Client program to connect to a database server, the program looks up the server name in the interfaces file and then connects to the server using the address.

The name, location, and contents of the interfaces file differ between operating systems. Also, the format of the addresses in the interfaces file differs between network protocols.

When you install Sybase IQ, the setup program creates a simple interfaces file that you can use for local connections to Sybase IQ over TCP/IP. It is the System Administrator's responsibility to modify the interfaces file and distribute it to users so that they can connect to Sybase IQ over the network.

## **Using the DSEDITION utility**

The Directory Services Editor (DSEDITION) is an Open Client utility that allows you to configure the interfaces file (*sql.ini* or *interfaces*). The following sections explain how to use the DSEDITION utility to configure the interfaces file. You must be the owner of the Sybase home directory (\$SYBASE on UNIX or %SYBASE% on Windows) in order to run DSEDITION.

Open Client, including the DSEDITION utility, is not installed with the Sybase IQ Network Client. For this reason you can only run DSEDITION where the IQ server is installed.

These sections describe how to use DSEDITION for those tasks required for Sybase IQ. It is not complete documentation for the DSEDITION utility. For more information on DSEDITION, see the *Utility Programs* book for your platform, included with other Sybase products.

## **Starting DSEDITION**

The *dsedit* executable is held in the *SYBASE\bin* directory, which is added to your path on installation. You can start DSEDITION either from the command line or (Windows only) by double-clicking *dsedit.exe* from the Windows Explorer

When you start DSEdit, the Select Directory Service window appears.



## Opening a Directory Services session

The Select Directory Service window allows you to open a session with a directory service. You can open a session to edit one of the following:

- Any directory service that has a driver listed in the *libtcl.cfg* file
- The interfaces file (*sql.ini*).

### ❖ Opening a session

- Select Interfaces Driver from the DS Name box and click OK.

---

**Note** The DSEdit utility uses the SYBASE environment variable to locate the *libtcl.cfg* file. If the SYBASE environment variable is not set correctly, DSEdit cannot locate the *libtcl.cfg* file.

---

You can add, modify, or delete entries for servers, including Sybase IQ servers, in this window.

## Adding a server entry

### ❖ Adding a server entry

- 1 Choose Add from the Server Object menu. The Input Server Name window appears.
- 2 Type a server name in the Server Name box, and click OK to enter the server name.

The server entry appears in the Server box. To specify the attributes of the server, you must modify the entry.

Server entry name  
need not match server  
command-line name

The server name entered here does not need to match the name provided on the Sybase IQ command line. The server *address*, not the server name, is used to identify and locate the server.

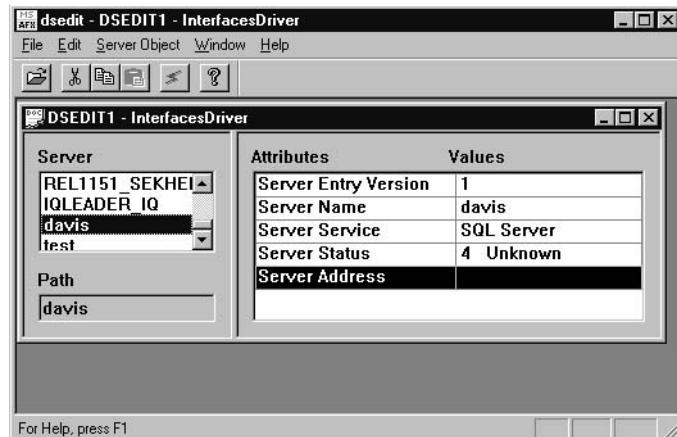
The server name field is purely an identifier for Open Client. For Sybase IQ, if the server has more than one database loaded, the DSEdit server name entry identifies which database to use.

## Adding or changing the server address

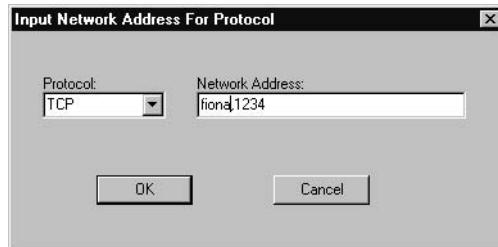
Once you have entered a Server Name, you need to modify the Server Address to complete the interfaces file entry.

### ❖ Entering a Server Address

- 1 Select a server entry in the Server box.
- 2 Select the Server Address in the Attributes box.



- 3 Double-click on the Server Address or right click and choose Modify Attribute from the popup menu. The Network Address Attribute window appears, showing the current value of the address. If you have no address entered, the box will be empty.
- 4 Click Add. The Input Network Address for Protocol window appears. Select TCP from the Protocol list box and enter a value in the Network Address text box.



For TCP/IP, addresses take one of the following two forms:

- computer name, port number
- IP-address, portnumber

The address or computer name is separated from the port number by a comma.

#### Machine name

The machine on which the server is running is identified by a name or an IP address. On Windows systems, you can find the machine name in Network Settings, in the Control Panel.

If your client and server are on the same machine, you must still enter the machine name. In this case, you can use

`localhost`

to identify the current machine.

#### Port Number

The port number you enter must match the port specified on the Sybase IQ database server command line, as described in “Starting the database server as an Open Server”. The default port number for Sybase IQ servers is 2638.

The following are valid server address entries:

```
elora,2638
123.85.234.029,2638
```

## Verifying the server address

You can verify your network connection by using the Ping command from the Server Object menu.

---

**Note** Verifying a network connection confirms that a server is receiving requests on the machine name and port number specified. It does not verify anything about database connections.

---

### ❖ Pinging a server

- 1 Ensure that the database server is running.
- 2 Click the server entry in the Server box of the *dsedit* session window.
- 3 Select Ping Server from the Server Object menu. The Ping window appears.
- 4 Click the address that you want to ping. Click Ping.  
A message box appears, to notify you if the connection is successful or not.  
A message box for a successful connection states that both Open Connection and Close Connection succeeded.

## Renaming a server entry

You can rename server entries from the *dsedit* session window.

### ❖ Renaming a server entry

- 1 Select a server entry in the Server box.
- 2 Choose Rename from the Server Object menu. The Input Server Name window appears.
- 3 Type a new name for the server entry in the Server Name box. Click OK to make the change.

## Deleting server entries

You can delete server entries from the *dsedit* session window.

### ❖ Deleting a server entry

- 1 Click a server entry in the Server box.
- 2 Choose Delete from the Server Object menu.

## Sybase applications and Sybase IQ

The ability of Sybase IQ to act as an Open Server enables Sybase applications such as OmniConnect to work with Sybase IQ. Note that, *in order to use the Open Client libraries, the client application must use only the supported system tables, views and stored procedures.*

### OmniConnect support

Sybase OmniConnect provides a unified view of disparate data within an organization, allowing users to access multiple data sources without having to know what the data looks like or where it is located. In addition, OmniConnect performs heterogeneous joins of data across the enterprise, enabling cross-platform table joins of targets such as DB2, Sybase Adaptive Server Enterprise, Adaptive Server Anywhere, Oracle, and VSAM.

Using the Open Server interface, Sybase IQ can act as a data source for OmniConnect.

## Open Client applications and Sybase IQ

You can build Open Client applications to access data in Sybase IQ base tables using the Open Client libraries directly from a C or C++ programming environment such as Powersoft Power++. If such applications reference catalog tables, views, or system stored procedures, these objects *must* be supported by *both* Adaptive Server Enterprise (Transact-SQL syntax) and Sybase IQ. The appendix “Transact-SQL Compatibility,” in the *Sybase IQ Reference Manual* describes how to create compatible applications.

## Setting up Sybase IQ as an Open Server

This section describes how to set up an Sybase IQ server to receive connections from Open Client applications.

## System requirements

There are separate requirements at the client and server for using Sybase IQ as an Open Server.

Server-side requirements At the server side, in order to use Sybase IQ as an Open Server, you must have a TCP/IP protocol stack.

---

**Note** When connecting to a remote Sybase IQ from a local Adaptive Server Enterprise server using OmniConnect, use these server classes:

- To connect to Sybase IQ 12.x, use server classes `asaodbc` and `asajdbc`.
  - To connect to Sybase IQ 11.x, use server class `asiq`.
- 

Client-side requirements In order to use Sybase client applications to connect to an Open Server, including Sybase IQ, you need the following:

- *Open Client components*—The Open Client libraries provide the network libraries that your application needs to communicate via TDS.
- *DSEdit*—The Directory Services Editor makes server names available to your Open Client application.

## Starting the database server as an Open Server

If you wish to use Sybase IQ as an Open Server, you must ensure that it is started using the TCP/IP protocol. By default, all available communications protocols are started by the server, but you can limit the protocols started by listing them explicitly on the command line. For example, the following command lines are both valid:

```
start_asiq -x tcpip,ipx other_server_switches  
asiqdemo.db  
start_asiq -x tcpip -n myserver other_server_switches  
asiqdemo.db
```

The first command line uses both TCP/IP and IPX protocols, of which TCP/IP is available for use by Open Client applications. The second line uses only TCP/IP.

The server can serve other applications through the TCP/IP protocol or other protocols using the Sybase IQ-specific application protocol at the same time as serving Open Client applications over TDS.

Port numbers Every application using TCP/IP on a machine uses a distinct TCP/IP **port**, so that network packets end up at the right application. The default port for Sybase IQ is port 2638, which is used for shared memory communications. You can specify a different port number with the Port network option:

```
start_asiq -x tcpip(Port=2629) -n myserver asiqdemo.db
```

- Open Client settings** To connect to this server, the *interfaces* file at the client machine must contain an entry specifying the machine name on which the database server is running, and the TCP/IP port it uses.
- For details on setting up the client machine, see “Configuring IQ Servers with DSEdit”.

## Configuring your database for use with Open Client

Your database must be Sybase IQ 12.0 or higher.

- Ensure your database is compatible** If you are using Sybase IQ together with Adaptive Server Enterprise, you should ensure that your database is created for maximum compatibility with Adaptive Server Enterprise.
- When connecting to Sybase IQ as an Open Server, applications frequently assume services they expect under Adaptive Server Enterprise (or SQL Server) are provided. These services are not always present.
- For information on creating Adaptive Server Enterprise-compatible databases, see Appendix A, “Compatibility with Other Sybase Databases,” in the *Sybase IQ Reference Manual*.

## Characteristics of Open Client and jConnect connections

When Sybase IQ is serving applications over TDS, it automatically sets relevant database options to values that are compatible with Adaptive Server Enterprise default behavior. These options are set temporarily, for the duration of the connection only. They can be overridden by the client application at any time.

- Default settings** The database options that are set on connection using TDS are as follows:

Option	Set to
ALLOW_NULLS_BY_DEFAULT	OFF
ANSINULL	OFF
AUTOMATIC_TIMESTAMP	ON
CHAINED	OFF

Option	Set to
CONTINUE_AFTER_RAISERROR	ON
DATE_FORMAT	YYYY-MM-DD
DATE_ORDER	MDY
ESCAPE_CHARACTER	OFF
ISOLATION_LEVEL	1
FLOAT_AS_DOUBLE	ON
QUOTED_IDENTIFIER	OFF
TIME_FORMAT	HH:NN:SS.SSS
TIMESTAMP_FORMAT	YYYY-MM-DD HH:NN:SS.SSS
TSQL_HEX_CONSTANT	ON
TSQL_VARIABLES	ON

How the startup options are set

The default database options are set for TDS connections using a system procedure named `sp_tsql_environment`. This procedure sets the following options:

```
SET TEMPORARY OPTION TSQL_VARIABLES='ON';
SET TEMPORARY OPTION ANSI_BLANKS='ON';
SET TEMPORARY OPTION TSQL_HEX_CONSTANT='ON';
SET TEMPORARY OPTION CHAINED='OFF';
SET TEMPORARY OPTION QUOTED_IDENTIFIER='OFF';
SET TEMPORARY OPTION ALLOW_NULLS_BY_DEFAULT='OFF';
SET TEMPORARY OPTION AUTOMATIC_TIMESTAMP='ON';
SET TEMPORARY OPTION ANSINULL='OFF';
SET TEMPORARY OPTION CONTINUE_AFTER_RAISERROR='ON';
SET TEMPORARY OPTION FLOAT_AS_DOUBLE='ON';
SET TEMPORARY OPTION ISOLATION_LEVEL='1';
SET TEMPORARY OPTION DATE_FORMAT='YYYY-MM-DD';
SET TEMPORARY OPTION TIMESTAMP_FORMAT='YYYY-MM-DD
HH:NN:SS.SSS';
SET TEMPORARY OPTION TIME_FORMAT='HH:NN:SS.SSS';
SET TEMPORARY OPTION DATE_ORDER='MDY';
SET TEMPORARY OPTION ESCAPE_CHARACTER='OFF'
```

---

**Note** Do not edit the `sp_tsql_environment` procedure yourself. It is for system use only. Options that are not supported by Sybase IQ are ignored.

---

The procedure only sets options for connections that use the TDS communications protocol. This includes Open Client and JDBC connections using jConnect. (Note that jConnect sets QUOTED\_IDENTIFIER to ON during its login sequence.) Other connections (ODBC and Embedded SQL) have the default settings for the database.

The sp\_iq\_process\_login system procedure, which is the default setting for the LOGIN\_PROCEDURE option, performs Sybase IQ User Administration and then calls sp\_login\_environment, which in turn calls sp\_tsql\_environment for TDS connections.

---

**Note** ODBC applications, including Interactive SQL applications, automatically set certain database options to values mandated by the ODBC specification. This overwrites settings by the LOGIN\_PROCEDURE database option. For details and a workaround, see “LOGIN\_PROCEDURE option,” *Sybase IQ Reference Manual*.

---

You can change the options for TDS connections and still use the Sybase IQ User Administration facility by calling a different procedure instead of sp\_login\_environment, as follows:

❖ **Changing the option settings for TDS connections**

- 1 Create a procedure that sets the database options you want. For example, you could use a procedure such as the following:

```
CREATE PROCEDURE my_startup_procedure()
BEGIN
    IF connection_property('CommProtocol')='TDS' THEN
        SET TEMPORARY OPTION QUOTED_IDENTIFIER='OFF';
    END IF
END
```

This procedure changes only the QUOTED\_IDENTIFIER option from the default settings.

- 2 To use the Sybase IQ User Administration feature, make a copy of the sp\_iq\_process\_login procedure, which is found in your \$ASDIR/scripts directory. Then edit sp\_iq\_process\_login to call your new procedure instead of sp\_login\_environment. The text of sp\_iq\_process\_login is found in your \$ASDIR/scripts directory. The line you must edit is:

```
call sp_login_environment();
```

---

**Note** Never edit the original, installed system procedures.

---

- 3 Future connections will use the procedure.

If you do not wish to use the Sybase IQ User Administration facility, you can skip step 2 and instead set the LOGIN\_PROCEDURE option to the name of a new procedure:

```
SET OPTION LOGIN_PROCEDURE= 'dba.my_startup_procedure'
```

For more information about database options, see Chapter 2, “Database Options” in the *Sybase IQ Reference Manual*.

#### Data type mappings

If you are developing Open Client applications, you should be aware of mappings between the data types supported by Sybase IQ and those expected by Open Client. For more information about these data type mappings, see the chapter entitled “The Open Client Interface” in *Adaptive Server Anywhere Programming Interfaces Guide*.

## Servers with multiple databases

Using Open Client Library, you can now connect to a specific database on a server with multiple databases.

- Set up entries in the *interfaces* file for the server.
- Use the -n parameter on the start\_asiq command to set up a shortcut for the database name.
- Specify the -S *database\_name* parameter with the database name on the isql command. *This parameter is now required whenever you connect.*

You can run the same program against multiple databases without changing the program itself by putting the shortcut name into the program and merely changing the shortcut definition.

For example, the following *interfaces* file excerpt defines two servers, *live\_sales* and *test\_sales*:

```
live_sales
    query tcp ether host8832 5555
    master tcp ether host 8832 5555
test_sales
    query tcp ether host8832 7777
    master tcp ether host 8832 7777
```

Start the server(s) and set up an alias for a particular database. The following command sets *live\_sales* equivalent to *salesbase.db*:

```
start_asiq -n sales_live <other parameters> \ -x
```

```
'tcpip{port=5555}' salesbase.db -n live_sales
```

To connect to the `live_sales` server, use this syntax:

```
isql -Udba -Psql -Slive_sales
```

A server name may only appear once in the *interfaces* file. Because the connection to Sybase IQ is now based on the database name, the database name must be unique. If all your scripts are set up to work on `salesbase` database, you will not have to modify them to work with `live_sales` or `test_sales`.



# Accessing Remote Data

## About this chapter

Sybase IQ can access data located on different servers, both Sybase and non-Sybase, as if the data were stored on the local server.

This chapter describes how to configure Sybase IQ to access remote data.

## Contents

Topic	Page
Sybase IQ and remote data	694
Multiplex servers and remote data access	712
Transaction management and remote data	713
Internal operations	714
Troubleshooting remote data access	718

## Sybase IQ and remote data

Using Sybase IQ you can

- Access data in Sybase databases, such as Adaptive Server Anywhere and Adaptive Server Enterprise.
- Access data in other relational databases such as Oracle and DB2.
- Access desktop data such as Excel spreadsheets, MS-Access databases, FoxPro, and text files.
- Access any other data source that supports an ODBC interface.
- Perform joins between local and remote data, although performance is much slower than if all the data is in a single IQ database).
- Perform joins between tables in separate Sybase IQ databases.
- Use certain Sybase IQ features on data sources that would normally not have that ability. Sybase IQ compensates for features not supported by a remote data source by operating on the data after it is retrieved.
- Use Sybase IQ to move data from one location to another using `INSERT SELECT`.
- Access remote servers directly using passthrough mode.
- Execute remote procedure calls to other servers.

Sybase IQ allows access to the following external data sources:

Accessible from  
Windows and UNIX

- Sybase IQ
- Adaptive Server Anywhere
- Adaptive Server Enterprise

Accessible from  
Windows only

- Oracle
- IBM DB2
- Microsoft SQL Server
- Other ODBC data sources

## Requirements for accessing remote data

This section describes the basic elements required to access remote data.

## Remote table mappings

Sybase IQ presents tables to a client application as if all the data in the tables were stored in the database to which the application is connected. Internally, when Sybase IQ executes a query involving remote tables, it determines the storage location and accesses the remote location to retrieve data.

To have remote tables appear as local tables to the client, you create local **proxy tables** that map to the remote data.

### ❖Creating a proxy table

- 1 Define the server where the remote data is located. This specifies the type of server and location of the remote server. For more information, see “Working with remote servers”.
- 2 Map the local user login information to the remote server user login information if the logins on the two servers are different. For more information, see “Working with external logins”.
- 3 Create the proxy table definition. This specifies the mapping of a local proxy table to the remote table. This includes the server where the remote table is located, and the database name, owner name, table name, and column names of the remote table. For more information, see “Working with proxy tables”.

Administering remote table mappings

You can manage remote table mappings and remote server definitions with the Sybase Central GUI or execute SQL statements directly with Interactive SQL.

## Server classes

A **server class** is assigned to each remote server. The server class specifies the access method used to interact with the server. Different types of remote servers require different access methods. The server classes provide Sybase IQ detailed server capability information. Sybase IQ adjusts its interaction with the remote server based on those capabilities.

JDBC-based server classes

There are currently two groups of server classes. The first is JDBC-based; the second is ODBC-based.

The JDBC-based server classes are:

- **asajdbc** for Sybase IQ Version 12 and later, and Adaptive Server Anywhere Version 6 and later
- **asejdbc** for Adaptive Server Enterprise and SQL Server Version 10 and later

	The JDBC-based server classes are the only classes that can be used with 64-bit UNIX IQ servers.
ODBC-based server classes	You must create a Data Source Name (DSN) before you create an ODBC-based server class.

The ODBC-based server classes supported on all platforms:

ODBC-based server class	Windows	UNIX
asaodbc for Sybase IQ Version 12 and later, and Adaptive Server Anywhere Version 5.5 and later	Yes	Yes
aseodbc for Adaptive Server Enterprise and SQL Server Version 10 and later (on Windows and 32-bit Linux platforms only)	Yes	
db2osbc for IBM DB2	Yes	
mssodbc for Microsoft SQL Server	Yes	
oraodbc for Oracle servers (version 8.0 and later)	Yes	
odbc for all other ODBC data sources	Yes	

For a full description of remote server classes, see Chapter 17, “Server Classes for Remote Data Access.”

## Working with remote servers

Before you can map remote objects to a local proxy table, you must define the remote server where the remote object is located. Defining a remote server, adds an entry to the sysservers table for the remote server. This section describes how to create, alter, and delete a remote server definition.

### Creating remote servers

Use the CREATE SERVER statement to set up remote server definitions.

For some systems, including Sybase IQ and Adaptive Server Anywhere, each data source describes a database, so a separate remote server definition is needed for each database.

You must have RESOURCE authority to create a server.

JDBC-based server example

The following statement creates an entry in the sysservers table for the Adaptive Server Enterprise server named ASEserver:

```
CREATE SERVER ASEserver  
CLASS 'asejdbc'
```

```
USING 'rimu:6666/my_asedb'
```

where:

- **ASEserver** is the name of the remote server
- **asejdbc** specifies the server is an Adaptive Server Enterprise and the connection to it is JDBC-based
- **rimu:6666/my\_asedb** is the machine name and TCP/IP port number where the remote server is located, and the database you will connect to. Specifying the database is optional when creating remote Adaptive Server Enterprise servers.

#### ODBC-based server example

For ODBC connections, each remote server corresponds to an ODBC data source.

ODBC-based connections may only be used for IQ on 32-bit systems. On 64-bit systems, other methods of access are available, described in sections, “Loading remote data without native classes” and “Querying data without native classes.”

To create an ODBC-based server class, you must first create a DSN, as described in “Creating and editing ODBC data sources” on page 97.

The following statement creates an entry in the `sysservers` table for the ODBC-based Sybase IQ server named `testasiq`:

```
CREATE SERVER testasiq
CLASS 'asaodbc'
USING 'test4'
```

where:

- **testasiq** is the name by which the remote server is known within this database.
- **asaodbc** specifies that the server is an Sybase IQ (or Adaptive Server Anywhere) and the connection to it uses ODBC.
- **test4** is the ODBC data source name.

For more information on server classes, including `CREATE SERVER` requirements for various server classes, see Chapter 17, “Server Classes for Remote Data Access.”

### Loading remote data without native classes

You need to use DirectConnect to access remote data sources:

- On 64-bit UNIX platforms
- On 32-bit platforms where no ODBC driver is available (for example, MS SQL Server)

This section and the following one provide examples of loading and querying data by means of DirectConnect.

**Non-Sybase remote data example**

For this example, assume the following:

- An Enterprise Connect Data Access (ECDA) server exists named *mssql* on UNIX host *monroe*, port 12530.
- The data is to be retrieved from an MS SQL server named *2000* on host *hardscrabble*, port 1433.

❖ **Loading MS SQL Server data into an IQ server on UNIX**

- 1 Using DirectConnect documentation, configure DirectConnect for your data source.
- 2 Make sure that ECDA server (*mssql*) is listed in the IQ interfaces file:

```
mssql
master tcp ether monroe 12530
query tcp ether monroe 12530
```

- 3 Add a new user, using the user id and password for server *mssql*:

```
isql -Udba -Psql -Stst_asiqdemo
grant connect to chill identified by chill
grant dba to chill
```

- 4 Log in as the new user to create a local table on IQ:

```
isql -Uchill -Pchill -Stst_asiqdemo
create table billing(status char(1), name
varchar(20), telno int)
```

- 5 Insert data:

```
insert into billing location 'mssql.pubs' { select *
from billing }
```

## Querying data without native classes

Currently the best approach to accessing non-Sybase data on 64-bit systems is to do so indirectly, as follows:

- 1 Configure ASE/CIS with a remote server and proxy to connect via DirectConnect. For example, use DirectConnect for Oracle to the Oracle server.
- 2 Configure IQ with a remote server using the ASEJDBC class to the ASE server. (The ASEODBC class is unavailable because there is no 64-bit Unix ODBC driver for ASE.)
- 3 Use the CREATE EXISTING TABLE statement to create proxy tables pointing to the proxy tables in ASE which in turn point to Oracle.

Querying remote data using DirectConnect and proxy table from UNIX

This example shows how to access MS SQL Server data. For this example, assume the following:

- A Sybase IQ server on host *sunbeam*, port 7594.
- An Adaptive Server Enterprise server on host *jones*, port 4101.
- An Enterprise Connect Data Access (ECDA) server exists named *mssql* on host *monroe*, port 12530.
- The data is to be retrieved from an MS SQL server named *2000* on host *hardscrabble*, port 1433.

❖ **Setting up Adaptive Server Enterprise for querying MS SQL Server**

- 1 Set up ASE and Component Integration Services (CIS) to MS SQL Server through DirectConnect. For example, assume that the server name is *jones\_1207*.
- 2 Add an entry to the ASE interfaces file to connect to *mssql*:

```
mssql  
master tcp ether monroe 12530  
query tcp ether monroe 12530
```

- 3 Enable CIS and remote procedure call handling from the ASE server. For example, if CIS is already enabled as the default,

```
sp_configure 'enable cis'
```

```
Parameter Name Default Memory Used Config Value Run Value  
enable cis      1      0          1          1  
(1 row affected)  
(return status=0)
```

```
sp_configure 'cis rpc handling', 1
```

```
Parameter Name Default Memory Used Config Value Run Value
```

```
enable cis      0      0          0      1
```

(1 row affected)

Configuration option changed. The SQL Server need not be rebooted since the option is dynamic.

You may need to reboot Adaptive Server Enterprise server after enabling CIS remote procedure call handling in older versions such as Sybase IQ 12.5.

- 4 Add the DirectConnect server to the ASE server's SYSSERVERS system table.

```
sp_addserver mssql, direct_connect, mssql  
Adding server 'mssql', physical name 'mssql'  
Server added.  
(Return status=0)
```

- 5 Create the user in Adaptive Server Enterprise that will be used in Sybase IQ to connect to ASE.

```
sp_addlogin tst, ttttst  
Password correctly set.  
Account unlocked. New login created.  
(return status = 0)  
  
grant role sa_role to tst  
use tst_db  
sp_adduser tst  
  
New user added.  
(return status = 0)
```

- 6 Add an external login from the master database:

```
use master  
sp_adDEXTERNLOGIN mssql, tst, chill, chill  
  
User 'tst' will be known as 'chill' in remote server  
'mssql'.  
(return status = 0)
```

- 7 Create an ASE proxy table as the added user from the desired database:

```
isql -Ust -Ttttst  
use test_db  
create proxy_table billing_tst at  
'mssql.pubs..billing'  
select * from billing_tst
```

```

status      name        telno
-----      -----
D           BOTANICALLY 1
B           BOTANICALL  2
(2 rows affected)

```

❖ **Setting up Sybase IQ to connect to the ASE server**

- 1 Add an entry to the IQ interfaces file:

```

jones_1207
master tcp ether jones 4101
query tcp ether jones 4101

```

- 2 Create the user to connect to ASE:

```

grant connect to tst identified by ttttst
grant dba to tst

```

- 3 Log in as the added user to create the ‘asejdbc’ server class and add external login:

```

isql -Utst -Ptsttst -Stst_asiqdemo
create SERVER jones_1207 CLASEE 'asejdbc' USING
'jones:4101/tst_db'
create existing table billing_iq at
'jones_4101.tst_db..billing_txt'
select * from billing_iq

status      name        telno
-----      -----
D           BOTANICALLY 1
B           BOTANICALL  2
(2 rows affected)

```

## **Deleting remote servers**

To drop a remote server from the Sybase IQ system tables, first drop all remote tables defined on that server, then use the DROP SERVER statement.

You must have DBA authority to delete a remote server.

### Example

❖ **Deleting a remote server (SQL)**

- 1 Connect to the host database from Interactive SQL using the jConnect driver (for JDBC connectivity).
- 2 Execute a DROP SERVER statement.

The following statement drops the server named testasiq:

```
DROP SERVER testasiq
```

For a full description of the DROP SERVER statement, see *Sybase IQ Reference Manual*.

## Altering remote servers

Use the ALTER SERVER statement to modify the attributes of a server. These changes do not take effect until the next connection to the remote server.

You must have RESOURCE authority to alter a server.

### Example

The following statement changes the server class of the server named ASEserver to aseodbc:

```
ALTER SERVER ASEserver  
CLASS 'aseodbc'
```

The Data Source Name for the server is ASEserver.

The ALTER SERVER statement can also be used to enable or disable a server's known capabilities.

For a complete description of the ALTER SERVER statement, see *Sybase IQ Reference Manual*.

## Listing the remote tables on a server

It may be helpful when you are configuring your Sybase IQ to get a list of the remote tables available on a particular server. The sp\_remote\_tables procedure returns a list of the tables on a server.

```
sp_remote_tables [servername]  
[,tablename]  
[, owner ]  
[, database]
```

If *tablename*, *owner*, or *database* is given, the list of tables is limited to only those that match.

For example, to get a list of all of the tables in the production database in an ASE named asetest, owned by 'fred':

```
sp_remote_tables asetest, null, fred, production
```

For more information, see the sp\_remote\_tables system procedure in *Sybase IQ Reference Manual*.

## **Listing remote server capabilities**

The `sp_servercaps` procedure displays information about a remote server's capabilities. Sybase IQ uses this capability information to determine how much of a SQL statement can be passed to a remote server.

The system tables that contain server capabilities are not populated until after Sybase IQ first connects to the remote server. This information comes from the `SYSCAPABILITY` and `SYSCAPABILITYNAME` system tables. The server name specified must be the same server name used in the `CREATE SERVER` statement.

Run the stored procedure `sp_servercaps` as follows:

```
sp_servercaps    servername
```

For more information, see the `sp_servercaps` system procedure in *Sybase IQ Reference Manual*.

## **Working with external logins**

Sybase IQ uses the names and passwords of its clients when it connects to a remote server on behalf of those clients. However, this behavior can be overridden by creating external logins. External logins are alternate login names and passwords that are used when communicating with a remote server.

When Sybase IQ connects to the remote server, `INSERT...LOCATION` uses the remote login for the user ID of the current connection, if a remote login has been created with `CREATE EXTERNLOGIN` and the remote server has been defined with a `CREATE SERVER` statement. If the remote server is not defined or a remote login has not been created for the user ID of the current connection, IQ connects using the user ID and password of the current connection. For an example of `INSERT...LOCATION` using a remote login, see “`INSERT` statement” in Chapter 6, “SQL Statements” of the *Sybase IQ Reference Manual*.

If you are using an integrated login, then the IQ name and password of the IQ client is the same as the database userid and password that the IQ userid maps to in `syslogins`.

## **Creating external logins**

Only the login-name and the DBA account can add or modify an external login.

The following statement allows the local user fred to gain access to the server ASEserver, using the remote login frederick with password banana.

```
CREATE EXTERNLOGIN fred  
TO ASEserver  
REMOTE LOGIN frederick  
IDENTIFIED BY banana
```

For more information, see CREATE EXTERNLOGIN statement in *Sybase IQ Reference Manual*.

## Dropping external logins

Use the DROP EXTERNLOGIN statement to remove external logins from the Sybase IQ system tables.

### Example

The following statement drops the external login for the local user fred created in the example above:

```
DROP EXTERNLOGIN fred TO ASEserver
```

For more information, see DROP EXTERNLOGIN statement in *Sybase IQ Reference Manual*.

## Working with proxy tables

Location transparency of remote data is enabled by creating a local proxy table that maps to the remote object. To create a proxy table you use one of the following statements:

- If the table already exists at the remote storage location, use the CREATE EXISTING TABLE statement. This statement defines the proxy table for an existing table on the remote server.

- If the table does not exist at the remote storage location, use the CREATE TABLE statement. This statement creates a new table on the remote server, and also defines the proxy table for that table.

---

**Note**

## Specifying proxy table locations

The AT keyword is used with both CREATE TABLE and CREATE EXISTING TABLE to define the location of an existing object. This location string has four components that are separated by either a period or a semicolon. Semicolons allow filenames and extensions to be used in the database and owner fields.

... AT '*server.database.owner.tablename*'

**Server**

The field *server* is the name by which the server is known in the current database, as specified in the CREATE SERVER statement. This field is mandatory for all remote data sources.

**Database**

The meaning of the *database* field depends on the data source. In some cases this field does not apply and should be left empty. The periods are still required, however.

- **Adaptive Server Enterprise** Specifies the database where the table exists, for example, master or pubs2.
- **Sybase IQ or Adaptive Server Anywhere** This field does not apply; leave it empty.

The database name for an Sybase IQ or Adaptive Server Anywhere ODBC data source should be specified when the data source name is defined in the ODBC Administrator.

For jConnect-based connections, the database should be specified in the USING clause of the CREATE SERVER statement.

For both ODBC and JDBC based connections to Sybase IQ or Adaptive Server Anywhere, you need a separate CREATE SERVER statement for each Sybase IQ or Anywhere database being accessed.

- **Excel, Lotus Notes, Access** For these file-based data sources, the database name is the name of the file containing the table. Since file names can contain a period, a semicolon should be used as the delimiter between server, database, owner, and table.

Owner	If the database supports the concept of ownership, the <i>owner</i> field represents the owner name. This field is only required when several owners have tables with the same name.
Tablename	The field <i>tablename</i> specifies the name of the table. In the case of an Excel spreadsheet, this is the name of the “sheet” in the workbook. If the table name is left empty, the remote table name is assumed to be the same as the local proxy table name. Tables used with Component Integration Services (CIS) cannot have names longer than 30 characters.

Examples:

- Sybase IQ:

```
'testasiq..DBA.employee'
```

- Adaptive Server Enterprise:

```
'ASEServer.pubs2.dbo.publishers'
```

- Excel:

```
'excel;d:\pcdb\quarter3.xls;;sheet1$'
```

- Access

```
'access;\\server1\production\inventory.mdb;;parts'
```

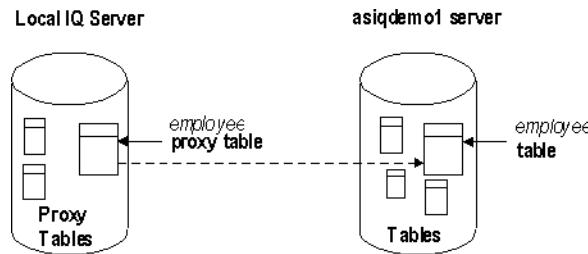
## Creating proxy tables

The CREATE EXISTING TABLE statement creates a proxy table that maps to an existing table on the remote server. Sybase IQ derives the column attributes and index information from the object at the remote location.

Example

To create a proxy table named p\_employee on the current server to a remote table named employee on the server named asiqdemo1, use the following syntax:

```
CREATE EXISTING TABLE p_employee  
AT 'asiqdemo1..DBA.employee'
```



For a full description of the CREATE EXISTING TABLE statement, see *Sybase IQ Reference Manual*.

## Using the CREATE TABLE statement

The CREATE TABLE statement creates a new table on the remote server, and defines the proxy table for that table when you use the AT option. You enter the CREATE TABLE statement using Sybase IQ data types. Sybase IQ automatically converts the data into the remote server's native types.

---

**Note** The CHAR data type is incompatible between Adaptive Server Anywhere and Sybase IQ when the database is built with BLANK PADDING OFF. If you want to perform cross-database joins between ASA and IQ tables using character data as the join key, use the CHAR data type with BLANK PADDING ON.

---

If you use the CREATE TABLE statement to create both a local and remote table, and then subsequently use the DROP TABLE statement to drop the proxy table, then the remote table also gets dropped. You can, however, use the DROP TABLE statement to drop a proxy table created using the CREATE EXISTING TABLE statement if you do not want to drop the remote table.

### Example

The following statement creates a table named employee on the remote server asiqdemo1, and creates a proxy table named members that maps to the remote location:

```
CREATE TABLE members
( membership_id INTEGER NOT NULL,
  member_name CHAR(30) NOT NULL,
  office_held CHAR( 20 ) NULL)
AT 'asiqdemo1..DBA.employee'
```

For more information, see CREATE TABLE statement in *Sybase IQ Reference Manual*.

## Listing the columns on a remote table

If you are entering a CREATE EXISTING TABLE statement and you are specifying a column list, it may be helpful to get a list of the columns that are available on a remote table. The sp\_remote\_columns system procedure produces a list of the columns on a remote table and a description of those data types.

```
sp_remote_columnsservername
```

```
[,tablename] [, owner ] [, database]
```

If a *tablename*, *owner*, or *database* name is given, the list of columns is limited to only those that match.

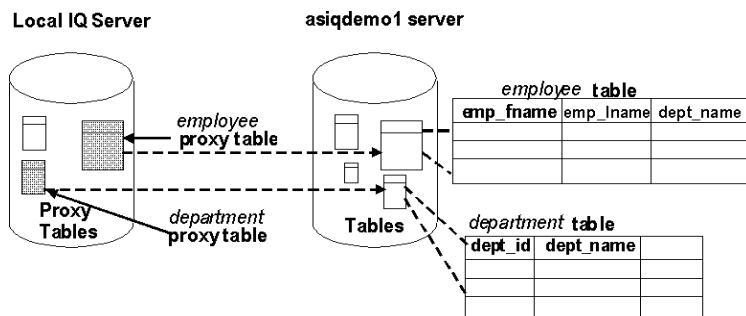
For example, to get a list of the columns in the sysobjects table in the production database in an Adaptive Server Enterprise server named asetest:

```
sp_remote_columns asetest, sysobjects, null,  
production
```

For more information, see the *sp\_remote\_columns* system procedure in the *Sybase IQ Reference Manual*.

## Example: a join between two remote tables

The following figure illustrates the remote Sybase IQ tables employee and department in the sample database mapped to the local server named testasiq.



This example shows how to:

- Define the remote testasiq server
- Create the proxy tables *employee* and *department*
- Perform a join between the remote *employee* and *department* tables.

In real-world cases, you may use joins between tables on different Sybase IQ databases. Here we describe a simple case using just one database, which may not be particularly useful, to illustrate the principles.

### ❖Performing a join between two remote tables, using Interactive SQL

- 1 Create a new database named *empty.db*.

This database holds no data. We will use it only to define the remote objects, and access the *asiqdemo* sample database from it.

- 2 Start a database server running both *empty.db* and the *asiqdemo* database. You can do this using the following command line, executed from the installation directory:

```
start_asiq asiqdemo empty
```

- 3 Connect to *empty.db* it from Interactive SQL using user ID DBA and password SQL.
- 4 In the new database, create a remote server named *testasiq*. Its server class is *asaodbc*, and the connection information is 'Sybase IQ Demo':

```
CREATE SERVER testasiq  
CLASS 'asaodbc' USING 'Sybase IQ Demo'
```

- 5 In this example, we use the same user ID and password on the remote database as on the local database, so no external logins are needed.
- 6 Define the *employee* proxy table:

```
CREATE EXISTING TABLE employee  
AT 'testasiq..DBA.employee'
```

- 7 Define the *department* proxy table:

```
CREATE EXISTING TABLE department  
AT 'testasiq..DBA.department'
```

- 8 Use the proxy tables in the SELECT statement to perform the join.

```
SELECT emp_fname, emp_lname, dept_name  
FROM employee JOIN department  
ON employee.dept_id = department.dept_id  
ORDER BY emp_lname
```

## Accessing multiple local databases

An Sybase IQ server may have several local databases running at one time. By defining tables in other local Sybase IQ databases as remote tables, you can perform cross database joins.

For example, if you are using database db1 and you want to access data in tables in database db2, you need to set up proxy table definitions that point to the tables in database db2. For instance, on an Sybase IQ named *testasiq*, you might have three databases available, db1, db2, and db3.

- If using ODBC, create an ODBC data source name entry for each database you will be accessing.

- Connect to one of the databases that you will be performing joins from. For example, connect to db1.
- Perform a CREATE SERVER for each other local database you will be accessing. This sets up a **loopback** connection to your Sybase IQ server.

```
CREATE SERVER local_db2
CLASS 'asaodbc'
USING 'testasiq_db2'
CREATE SERVER local_db3
CLASS 'asaodbc'
USING 'testasiq_db3'
```

or using JDBC:

```
CREATE SERVER local_db2
CLASS 'asajdbc'
USING 'mypc1:2638/db2'
CREATE SERVER local_db3
CLASS 'asajdbc'
USING 'mypc1:2638/db3'
```

- Create proxy table definitions using CREATE EXISTING TABLE to the tables in the other databases you want to access.

```
CREATE EXISTING TABLE employee
AT 'local_db2...employee'
```

## Sending native statements to remote servers

Use the FORWARD TO statement to send one or more statements to the remote server in its native syntax. This statement can be used in two ways:

- To send a statement to a remote server
- To place Sybase IQ into passthrough mode for sending a series of statements to a remote server

If a connection cannot be made to the specified server, the reason is contained in a message returned to the user. If a connection is made, any results are converted into a form that can be recognized by the client program.

The FORWARD TO statement can be used to verify that a server is configured correctly. If you send a statement to the remote server and Sybase IQ does not return an error message, the remote server is configured correctly.

### Example 1

The following statement verifies connectivity to the server named ASEserver by selecting the version string:

```
FORWARD TO ASEserver {SELECT @@version}
```

**Example 2**

The following statements show a passthrough session with the server named ASEserver:

```
FORWARD TO ASEserver  
select * from titles  
select * from authors  
FORWARD TO
```

The FORWARD TO statement without aservername turns off passthrough mode. For more information, see FORWARD TO statement in the *Sybase IQ Reference Manual*.

## Using remote procedure calls (RPCs)

Sybase IQ users can issue procedure calls to remote servers that support the feature.

Sybase Sybase IQ, Adaptive Server Anywhere, and Adaptive Server Enterprise, as well as Oracle and DB2, support this feature. Issuing a remote procedure call is similar to using a local procedure call.

### ❖ **Issuing a remote procedure call**

- 1 First define the procedure to Sybase IQ.

The syntax is the same as a local procedure definition except instead of using SQL statements to make up the body of the call, a location string is given defining the location where the procedure resides.

```
CREATE PROCEDURE remoteuser (IN uname char(30))  
AT 'bostonase.master dbo.sp_helpuser'
```

- 2 Execute the procedure as follows:

```
call remoteuser ('josh')
```

Here is another example:

```
CREATE PROCEDURE sp_tables_remote  
(IN table_name char(128))  
AT 'asiqidemo_svr...sp_tables'
```

You can execute the procedure either as

```
call sp_tables_remote ('employee')
```

or from DBISQL as

```
sp_tables_remote employee
```

Data types for remote procedures

The following data types are allowed for RPC parameters. Other data types are disallowed:

- [ UNSIGNED ] SMALLINT
- [ UNSIGNED ] INT
- [ UNSIGNED ] BIGINT
- TINYINT
- REAL
- DOUBLE
- CHAR
- CHAR
- BIT

NUMERIC and DECIMAL data types are allowed for IN parameters, but not for OUT or INOUT parameters.

To delete a remote procedure, connect to the database and execute a DROP PROCEDURE statement.

## Multiplex servers and remote data access

Sybase IQ supports remote data access among servers in a multiplex configuration in all directions:

- Write server to query server
- Query server to write server
- Query server to query server

For example, the write server may be defined as a remote server to a query server and proxy tables may be defined on the query server for tables on the write server.

Consider the following when using remote data access among multiplex servers:

- Use server class ASAODBC or ASA JDBC for IQ servers

- Use proxy table names that differ from local table names to avoid multiplex static collisions. For example, if a query server local store has a table named *employee*, then the proxy table name should not be *employee*.
- Add remote definitions created on the query servers to the `sp_mpvcfg_<servername>` stored procedure so that they persist after a multiplex SYNCHRONIZE. If you do not add the definitions to the `sp_iqmpvcfg_<servername>` procedure, they disappear after SYNCHRONIZE.

For more information on the `sp_mpvcfg_<servername>` procedure, please see “Synchronizing query servers” on page 204.
- A remote server definition that points to the current server returns an expected server definition is circular error if attempting to use that remote server definition.

## Transaction management and remote data

Transactions provide a way to group SQL statements so that they are treated as a unit—either all work performed by the statements is committed to the database, or none of it is.

Transaction management with remote tables is handled somewhat differently than it is for local IQ tables. Transaction management for remote tables is handled for the most part as it is in Adaptive Server Anywhere, although there are some differences.

For a general discussion of transactions in Adaptive Server Anywhere, see “Using Transactions and Locks” in the *Adaptive Server Anywhere User’s Guide*. For a general discussion of transactions in Sybase IQ, see Chapter 10, “Transactions and Versioning.”

## Remote transaction management overview

The method for managing transactions involving remote servers uses a **two-phase commit** protocol. Sybase IQ implements a strategy that ensures transaction integrity for most scenarios. However, when more than one remote server is invoked in a transaction, there is still a chance that a distributed unit of work will be left in an undetermined state. Even though two-phase commit protocol is used, no recovery process is included.

The general logic for managing a user transaction is as follows:

- 1 Sybase IQ prefaces work to a remote server with a BEGIN TRANSACTION notification.
- 2 When the transaction is ready to be committed, Sybase IQ sends a PREPARE TRANSACTION notification to each remote server that has been part of the transaction. This ensures that the remote server is ready to commit the transaction.
- 3 If a PREPARE TRANSACTION request fails, all remote servers are told to roll back the current transaction.  
If all PREPARE TRANSACTION requests are successful, the server sends a COMMIT TRANSACTION request to each remote server involved with the transaction.

Any statement preceded by BEGIN TRANSACTION can begin a transaction. Other statements are sent to a remote server to be executed as a single, remote unit of work.

## **Restrictions on transaction management**

Restrictions on transaction management are as follows:

- Savepoints are not propagated to remote servers.
- If nested BEGIN TRANSACTION and COMMIT TRANSACTION statements are included in a transaction that involves remote servers, only the outermost set of statements is processed. The innermost set, containing the BEGIN TRANSACTION and COMMIT TRANSACTION statements, is not transmitted to remote servers.

## **Internal operations**

This section describes the underlying operations on remote servers performed by Sybase IQ on behalf of client applications.

## Query parsing

When a statement is received from a client, it is parsed. An error is raised if the statement is not a valid Sybase IQ SQL statement.

## Query normalization

The next step is called query normalization. During this step, referenced objects are verified and some data type compatibility is checked.

For example, consider the following query:

```
SELECT *
FROM t1
WHERE c1 = 10
```

The query normalization stage verifies that table t1 with a column c1 exists in the system tables. It also verifies that the data type of column c1 is compatible with the value 10. If the column's data type is datetime, for example, this statement is rejected.

## Query preprocessing

Query preprocessing prepares the query for optimization. It may change the representation of a statement so that the SQL statement Sybase IQ generates for passing to a remote server will be syntactically different from the original statement.

Preprocessing performs view expansion so that a query can operate on tables referenced by the view. Expressions may be reordered and subqueries may be transformed to improve processing efficiency. For example, some subqueries may be converted into joins.

## Server capabilities

The previous steps are performed on all queries, both local and remote.

The following steps depend on the type of SQL statement and the capabilities of the remote servers involved.

Each remote server defined to Sybase IQ has a set of capabilities associated with it. These capabilities are stored in the syscapabilities system table. These capabilities are initialized during the first connection to a remote server. The generic server class odbc relies strictly on information returned from the ODBC driver to determine these capabilities. Other server classes such as those for Sybase products have detailed knowledge of the capabilities of a remote server type and use that knowledge to supplement what is returned from the driver.

Once syscapabilities is initialized for a server, the capability information is retrieved only from the system table. This allows a user to alter the known capabilities of a server.

Since a remote server may not support all of the features of a given SQL statement, Sybase IQ must break the statement into simpler components to the point that the query can be given to the remote server. SQL features not passed off to a remote server must be evaluated by Sybase IQ itself.

For example, a query may contain an ORDER BY statement. If a remote server cannot perform ORDER BY, the statement is sent to the remote server without it and Sybase IQ performs the ORDER BY on the result returned, before returning the result to the user. The result is that the user can employ the full range of Sybase IQ supported SQL without concern for the features of a particular back end.

## **Complete passthrough of the statement**

The most efficient way to handle a statement is usually to hand off as much of the original statement as possible to the remote server involved. Sybase IQ will attempt to pass off as much of the statement as is possible. In many cases this will be the complete statement as originally given to Sybase IQ.

Sybase IQ will hand off the complete statement when:

- Every table in the statement resides in the same remote server.
- The remote server is capable of processing all of the syntax in the statement.

In rare conditions, it may actually be more efficient to let Sybase IQ do some of the work instead of passing it off. For example, Sybase IQ may have a better sorting algorithm. In this case you may consider altering the capabilities of a remote server using the ALTER SERVER statement.

For more information see the ALTER SERVER statement in *Sybase IQ Reference Manual*.

If you are writing queries that access tables in more than one database and you would like to divide the workload between servers, refer to the description of query processing within CIS in the section “Query processing” of the chapter “Understanding Component Integration Services” in the *Component Integration Services User’s Guide for ASE and OmniConnect* for more information.

## Partial passthrough of the statement

If a statement contains references to multiple servers, or uses SQL features not supported by a remote server, the query is decomposed into simpler parts.

### Select

SELECT statements are broken down by removing portions that cannot be passed on and letting Sybase IQ perform the feature. For example, assume that a remote server cannot process the atan2() function in the following statement:

```
select acol, bcol, ccol
where atan2(bcol,10) > 3
and ccol = 10
```

The statement sent to the remote server is converted to:

```
select acol, bcol, ccol
where ccol = 10
```

Locally, Sybase IQ applies “where atan2(bcol,10) > 3” to the intermediate result set.

### Joins

Sybase IQ processes joins involving remote tables using a nested loop algorithm. When two tables are joined, one table is selected to be the outer table. The outer table is scanned based on the WHERE conditions that apply to it. For every qualifying row found, the other table, known as the inner table is scanned to find a row that matches the join condition.

Since the cost of searching a remote table is usually much higher than a local table (due to network I/O), every effort is made to make the remote table the outermost table in the join.

---

**Note** When you join one or more remote tables to one or more local IQ tables, the part of the join that includes local tables is resolved locally, taking advantage of any join indexes that exist for the joined local columns. However, the part of the join that involves the remote tables does not use join indexes. As a result, you will likely notice slower responses to queries that include joins between local and remote tables.

---

#### Update and delete

You can issue an UPDATE or DELETE on a proxy table for a remote server that supports updates, as long as the query can be passed off completely.

For example, you can send the following update to a remote Sybase IQ 12.4.3 or higher database:

```
UPDATE t1  
SET a = atan2(b, 10)  
WHERE b > 5
```

If Sybase IQ cannot pass off an UPDATE or DELETE completely, or if you send the query to a remote server that does not support updates, you receive an error. Sybase IQ does not support positioned updates on remote data.

---

**Note** An UPDATE or DELETE cannot be performed if an intermediate temporary table is required in Sybase IQ. This occurs in queries with ORDER BY and some queries with subqueries.

---

## Troubleshooting remote data access

This section provides some hints for troubleshooting remote servers.

## Features not supported for remote data

The following features are not supported on remote data. Some are never supported by Sybase IQ. Others are supported for local data only. Attempts to use these features on remote data will run into problems:

- ALTER TABLE statement is not supported against remote tables.
- Triggers defined on proxy tables will not fire.
- SQL Remote is not supported.
- Java data types are not supported.
- Foreign keys that refer to remote tables are ignored.
- The READTEXT, WRITETEXT, and TEXTPTPR functions are not supported.
- UPDATE and DELETE requiring an intermediate temporary table are not supported.
- Positioned UPDATE and DELETE are not supported.
- Backwards scrolling on cursors opened against remote data is not supported. Fetch statements must be NEXT or RELATIVE 1.
- If a column on a remote table has a name that is a keyword on the remote server, you cannot access data in that column. Sybase IQ cannot know all of the remote server reserved words. You can execute a CREATE EXISTING TABLE statement and import the definition, but you cannot select that column.
- When using Component Integration Services (CIS) in certain geographic regions, connection attempts return the error No Suitable Driver. Java Developer Kits used with Sybase IQ 12.7 support certain time zone codes. See “TZ environment variable,” Chapter 1, “File Locations and Installation Settings,” in the *Sybase IQ Reference Manual*.

## Performance limitations for Java Virtual Machine

Java applications running in IQ run slower than when run outside in a Sun JVM. Despite this limitation, Sybase recommends that you tune your applications by increasing the available memory for IQ JVM use with the database options JAVA\_HEAP\_SIZE and JAVA\_NAMESPACE\_SIZE.

## Case sensitivity

The case sensitivity setting of your IQ database should match the settings used by any remote servers accessed.

IQ databases are created case insensitive by default. With this configuration, unpredictable results may occur when selecting from a case sensitive database. Different results will occur depending on whether ORDER BY or string comparisons are pushed off to a remote server or evaluated by the local Sybase IQ.

## Connectivity problems

Take the following steps to be sure you can connect to a remote server:

- Determine that you can connect to a remote server using a client tool such as Interactive SQL (DBISQL) before configuring Sybase IQ.
- Perform a simple passthrough statement to a remote server to check your connectivity and remote login configuration. For example:

```
FORWARD TO testasiq {select @@version}
```

- Turn on remote tracing for a trace of the interactions with remote servers.

```
SET OPTION cis_option = 2
```

## General problems with queries

If you are faced with some type of problem with the way Sybase IQ is handling a query against a remote table, it is usually helpful to understand how Sybase IQ is executing that query. You can display remote tracing as well as a description of the query execution plan:

```
SET OPTION cis_option = 6
```

## Queries blocked on themselves

If you access multiple databases on a single Sybase IQ or Adaptive Server Anywhere server, you may need to increase the number of threads used by the database server using the -gx command-line switch. By default, this switch is set to one more than the number of CPUs on the machine.

You must have enough threads available to support the individual tasks that are being run by a query. Failure to provide the number of required tasks can lead to a query becoming blocked on itself.

---

**Note** The -gx switch is not documented in the *Sybase IQ Reference Manual*, as you do not normally need to set it for an IQ database. For any purpose other than the one described here, if you need to increase the number of threads you set the -iqmt switch, which controls the number of threads for IQ Store operations.

---

## Managing remote data access connections

If you access remote databases via ODBC, the connection to the remote server is given a name. The name can be used to drop the connection as one way to cancel a remote request.

The connections are named ASACIS\_ conn-name, where conn-name is the connection ID of the local connection. The connection ID can be obtained from the `sa_conn_info` stored procedure.



# Server Classes for Remote Data Access

## About this chapter

This chapter describes how Sybase IQ interfaces with various server classes.

## Contents

Topic	Page
Server classes overview	723
JDBC-based server classes	724
ODBC-based server classes	727

## Server classes overview

The behavior of a remote connection is determined by the server class in the CREATE SERVER statement. The server classes give Sybase IQ detailed server capability information. Sybase IQ formats SQL statements specific to a server's capabilities.

There are two categories of server classes:

- JDBC-based server classes
- ODBC-based server classes

Each server class has a set of unique characteristics that database administrators and programmers need to know about in order to configure the server for remote data access.

When using this chapter, refer both to the section generic to the server class category (JDBC-based or ODBC-based), and to the section specific to the individual server class.

## JDBC-based server classes

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

- **asajdbc** Sybase IQ Version 12 or later, and Adaptive Server Anywhere Version 6 and later
- **asejdbc** Adaptive Server Enterprise and SQL Server Version 10 and later.

## Configuration notes for JDBC classes

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

- Your local database must be enabled for Java. Do not specify JAVA OFF when you create the database if you plan to use a JDBC-based server class.
- The Java virtual machine needs more than the default amount of memory to load and run jConnect. Set these memory options to at least the following values:

```
SET OPTION "PUBLIC".JAVA_NAMESPACE_SIZE = 3000000  
SET OPTION "PUBLIC".JAVA_HEAP_SIZE = 1000000
```

- As long as jConnect 5.5 is installed with Sybase IQ, no additional drivers need to be installed.
- For optimum performance, Sybase recommends an ODBC-based class (asaodbc or aseodbc). See “Server class aseodbc” on page 729 for platform restrictions on the aseodbc class.
- Any remote server that you access using the asejdbc or asajdbc server class must be set up to handle a jConnect 5.5 based client. Setup occurs automatically for all supported Sybase servers.

## Server class asajdbc

A server with server class asajdbc is one of:

- Sybase IQ Version 12 or later
- Adaptive Server Anywhere Version 6 or later

No special requirements exist for the configuration of an Sybase IQ or Adaptive Server Anywhere data source.

## USING parameter value in the CREATE SERVER statement

The USING parameter in the CREATE SERVER statement takes the form *hostname:portnumber [/databasename]*, where:

- **hostname** is the machine that the remote server is running on
- **portnumber** is the TCP/IP port number that the remote server is listening on. The default port number that an Sybase IQ listens on is 2638.
- **databasename** is the Sybase IQ database that the connection will use. This is the name specified in the -n switch when the server was started, or in the DBN (DatabaseName) connection parameter.

For example, to configure an Sybase IQ named testasiq that is located on the machine apple and listening on port number 2638, use:

```
CREATE SERVER testasiq
CLASS 'asajdbc'
USING 'apple:2638'
```

You must issue a separate CREATE SERVER for each Sybase IQ or Adaptive Server Anywhere database that you intend to access. For example, if an Sybase IQ server named testasiq is running on the machine 'banana' and owns three databases (db1, db2, db3), you configure the local Sybase IQ like this:

```
CREATE SERVER testasiqdb1
CLASS 'asajdbc'
USING 'banana:2638/db1'
CREATE SERVER testasaiqdb2
CLASS 'asajdbc'
USING 'banana:2638/db2'
CREATE SERVER testasiqdb2
CLASS 'asajdbc'
USING 'banana:2638/db3'
```

If you do not specify a */databasename* value, the remote connection uses the remote Sybase IQ or Adaptive Server Anywhere default database. Since this may not be the database you want to connect to, you should always specify a */databasename* in the USING clause.

## Server class asejdbc

A server with server class asejdbc can be:

- Adaptive Server Enterprise
- SQL Server Version 10 and later

No special requirements exist for the configuration of an Adaptive Server Enterprise data source.

## Data type conversions

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

Sybase IQ data type	ASE default data type
bit	bit
tinyint	tinyint
smallint	smallint
int	int
integer	integer
decimal [defaults precision=30, scale=6]	numeric(30,6)
decimal(128,128)	not supported
numeric [defaults precision=30, scale=6]	numeric(30,6)
numeric(128,128)	not supported
float	real
real	real
double	float
smallmoney	numeric(10,4)
money	numeric(19,4)
date	datetime
time	datetime
timestamp	datetime
datetime	datetime
char(n)	char(n)
character(n)	char(n)
varchar(n)	varchar(n)

Sybase IQ data type	ASE default data type
character varying(n)	varchar(n)
binary(n)	binary(n)
varbinary(n)	varbinary(n)
text	text
bigint	numeric(20,0)

## ODBC-based server classes

Sybase IQ supports these ODBC-based server classes:

- asaodbc
- aseodbc
- db2odbc
- mssodbc
- oraodbc
- odbc

---

**Note** For 64-bit platforms, only the class asaodbc is supported.

---

## Defining ODBC external servers

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

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

For example, to configure an Adaptive Server Enterprise server named myase whose Data Source Name is also myase, use:

```
CREATE SERVER myase
CLASS 'aseodbc'
USING 'myase'
```

**Using connection strings instead of data sources**

For more information on creating ODBC data sources for Sybase IQ, see “Creating and editing ODBC data sources” on page 97. For information on creating ODBC data sources for Adaptive Server Enterprise, see “Server class aseodbc”.

If you do not wish to use data sources, you can supply a connection string in the USING clause of the CREATE SERVER statement.

To do this, you must know the connection parameters for the ODBC driver you are using. For example, a connection to an Sybase IQ may be as follows:

```
CREATE SERVER testasiq
CLASS 'asaodbc'
USING 'driver=adaptive server IQ 12.0;
eng=testasaiq;dbn=asiqdemo;links=tcpip{}'
```

This defines a connection to an Sybase IQ database server named testasiq, database asiqdemo, using the TCP/IP protocol.

**See also**

For information specific to particular ODBC server classes, see:

- “Server class asaodbc”
- “Server class aseodbc”

## **Server class asaodbc**

A server with server class asaodbc is one of:

- Sybase IQ Version 12 or later
- Adaptive Server Anywhere Version 5.5 or later

No special requirements exist for the configuration of an Adaptive Server Anywhere or Sybase IQ data source.

The ODBC driver for Sybase IQ 12 databases is installed when you install Sybase IQ.

To access Adaptive Server Anywhere version 5 or version 6 servers, install the appropriate ODBC driver (Version 5 driver for Version 5.5 databases, or Version 6 driver for Version 6 or later databases). You cannot use the Version 6 ODBC driver to connect to a Version 5 Adaptive Server Anywhere.

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

For 64-bit platforms, asaodbc is the only supported server class.

## Server class aseodbc

A server with server class aseodbc is:

- Adaptive Server Enterprise
- SQL Server (version 10 and later)

Sybase IQ requires the local installation of the Adaptive Server Enterprise ODBC driver and Open Client connectivity libraries to connect to a remote Adaptive Server with class aseodbc. However, the performance is better than with the asejdbc class.

### Notes

- Open Client should be version 11.1.1, EBF 7886 or above. Install Open Client and verify connectivity to the Adaptive Server before you install ODBC and configure Sybase IQ. The Sybase ODBC driver should be version 11.1.1, EBF 7911 or above.
- If the data resides on a 64-bit system, the aseodbc class is unavailable because no UNIX-based ODBC driver for ASE exists. Use the asejdbc class instead.
- Configure a User Data Source in the ODBC Data Source Administrator with the following attributes:
  - On the ODBC tab:  
Enter any value for Data source name. This value is used in the USING clause of the CREATE SERVER statement.
  - On the Database tab:  
Server name should match the name of the server in the Sybase *interfaces* file.
  - Under the Advanced tab, check the Application Using Threads box and check the Enable Quoted Identifiers box.
  - Under the Connection tab:  
Set the charset field to match your Sybase IQ character set.
  - Under the Performance tab:  
Set Prepare Method to “2-Full.”

Set Fetch Array Size as large as possible for best performance. This increases memory requirements since this is the number of rows that must be cached in memory. Sybase recommends using a value of 100.

Set Select Method to “0-Cursor.”

Set Packet Size to as large as possible. Sybase recommends using a value of -1.

Set Connection Cache to 1.

## Data type conversions

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

Sybase IQ data type	ASE default data type
bit	bit
tinyint	tinyint
smallint	smallint
int	int
integer	integer
decimal [defaults precision=30, scale=6]	numeric(30,6)
decimal(128,128)	not supported
numeric [defaults precision=30, scale=6]	numeric(30,6)
numeric(128,128)	not supported
float	real
real	real
double	float
smallmoney	numeric(10,4)
money	numeric(19,4)
date	datetime
time	datetime
timestamp	datetime
datetime	datetime
char(n)	char(n)
character(n)	char(n)
varchar(n)	varchar(n)

Sybase IQ data type	ASE default data type
character varying(n)	varchar(n)
binary(n)	binary(n)
varbinary(n)	varbinary(n)
text	text
bigint	numeric(20,0)

## Server class db2odbc

A server with server class db2odbc is IBM DB2.

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

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

## Data type conversions

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

Sybase IQ data type	DB2 default data type
bit	smallint
tinyint	smallint
smallint	smallint
int	int
integer	int
bigint	decimal(20,0)
decimal [defaults precision=30, scale=6]	decimal(30,6)
decimal(128,128)	not supported
numeric [defaults precision=30, scale=6]	decimal(30,6)
numeric(128,128)	not supported

Sybase IQ data type	DB2 default data type
float	float
real	real
double	float
smallmoney	decimal(10,4)
money	decimal(19,4)
date	date
time	time
smalldatetime	timestamp
timestamp	timestamp
datetime	timestamp
char(1-254)	varchar(n)
character(255-4000)	varchar(n)
char(4001-32767)	long varchar
varchar(1-4000)	varchar(n)
varchar(4001-32767)	long varchar
character varying(1-4000)	varchar(n)
character varying(4001-32767)	long varchar
binary(1-255)	varchar for bit data
varbinary (256-4000)	varchar for bit data
varbinary(4001-32767)	long varchar for bit data
long binary	long varchar for bit data
text	long varchar

## Server class oraodbc

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

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

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

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

Use only Sybase IQ servers *running on Windows* to access Oracle data. There is no 64-bit driver manager for Sybase IQ servers running on 64-bit UNIX systems.

❖ **Querying Oracle data using Sybase IQ servers on 64-bit UNIX**

This process tells how to create the two sets of proxy tables needed to query Oracle data in this situation.

- 1 Configure DirectConnect for Oracle to connect to Oracle.
- 2 Configure proxy tables in DirectConnect for Oracle.
- 3 Create a remote server in Sybase IQ using the ASEJDBC class to the server and the port number for DirectConnect for Oracle.
- 4 Use a CREATE EXISTING TABLE statement to create a proxy table pointing to the ASE proxy tables in DirectConnect for Oracle.

❖ **Loading Oracle data using Sybase IQ servers on 64-bit UNIX**

For best performance when loading large data, access the remote database with a different method from that used for queries, as follows:

- 1 Create proxy tables in DirectConnect for Oracle.
- 2 Use the INSERT .. LOCATION statement to the proxy tables.

For details about INSERT .. LOCATION, see “Inserting from a different database” on page 354.

## Data type conversions

When you issue a CREATE TABLE statement, Sybase IQ automatically converts the data types to the corresponding Oracle data types. The following table describes the Sybase IQ to Oracle data type conversions.

Sybase IQ data type	Oracle data type
bit	number(1,0)
tinyint	number(3,0)
smallint	number(5,0)
int, integer	number(11,0)
bigint	number(20,0)
decimal (precision, scale)	number(prec, scale)

Sybase IQ data type	Oracle data type
numeric (precision, scale)	number(prec, scale)
float	float
real	real
smallmoney	numeric(13,4)
money	number(19,4)
date	date
time	date
smalldatetime	date
timestamp	date
datetime	date
char(n)	if (n>255) long else varchar(n)
varchar(n)	if (n>2000) long else varchar(n)
binary(n)	if (n>255) long raw else raw(n)
varbinary(n)	if (n>255) long raw else raw(n)
long binary	long raw

## Server class mssqlodbc

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

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

```
CREATE EXISTING TABLE accounts
AT 'mymssql.database.owner.accounts'
```

## Data type conversions

When you issue a CREATE TABLE statement, Sybase IQ automatically converts the data types to the corresponding Microsoft SQL Server data types. The following table describes the Sybase IQ to Microsoft SQL Server data type conversions.

Sybase IQ data type	Microsoft SQL Server default data type
bit	bit
tinyint	tinyint
smallint	smallint
int, integer	int
bigint	numeric(20,0)
decimal [defaults precision=30, scale=6]	decimal(prec, scale)
numeric [defaults precision=30, scale=6]	numeric(prec, scale)
float	if (prec) float(prec) else float
real	real
smallmoney	smallmoney
money	money
date	datetime
time	datetime
smalldatetime	datetime
datetime	datetime
timestamp	datetime
char(n)	if (length>255) text else varchar(length)
character(n)	char(n)
varchar(n)	if (length>255) text else varchar(length)
binary(n)	if (length>255) image else binary(length)
long binary	image
double	float
uniqueidentifierstr	uniqueidentifier

## Server class odbc

ODBC data sources that do not have their own server class use server class `odbc`. You can use any ODBC driver that complies with ODBC version 2.0 compliance level 1 or higher. Sybase supports the following ODBC data sources:

- Microsoft Excel

- Microsoft Access
- Microsoft Foxpro
- Lotus Notes

The latest versions of Microsoft ODBC drivers can be obtained through the Microsoft Data Access Components (MDAC) distribution found at the Microsoft download Web site. The Microsoft driver versions listed below are part of MDAC 2.0.

The following sections provide notes on accessing these data sources.

## **Microsoft Excel (Microsoft 3.51.171300)**

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

In this example, an ODBC data source named excel was created. To create a workbook named work1.xls with a sheet (table) called mywork:

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

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

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

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

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

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

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

Adding the \$ (dollar sign) to the sheet name indicates that the entire worksheet should be selected.

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

Deletes are not supported. Also some updates may not be possible since neither the Excel driver nor IQ support positioned updates.

### **Microsoft Access (Microsoft 3.51.171300)**

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

Assuming an ODBC data source named access, create a table for it in one of these ways:

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

or

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

or

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

or

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

Access does not support the owner name qualification. Leave it empty.

### **Microsoft Foxpro (Microsoft 3.51.171300)**

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

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

This statement creates a file named *d:\pcdb\fox1.dbf* when you choose the “free table directory” option in the odbc driver manager.

## Lotus Notes SQL 2.0 (2.04.0203)

You can obtain this driver from the Lotus Web site. Read the documentation that comes with it for an explanation of how Notes data maps to relational tables. You can easily map IQ tables to Notes forms.

To set up Sybase IQ to access the Address sample file, follow this procedure.

### ❖ Setting up IQ to access the Address sample file

- 1 Create an ODBC data source using the NotesSQL driver.

The database will be the sample names file *c:\notes\data\names.nsf*. The Map Special Characters option should be turned on. For this example, the Data Source Name is *my\_notes\_dsn*.

- 2 Create an IQ server:

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

- 3 Map the Person form into an IQ table:

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

- 4 Query the table

```
SELECT * FROM Person
```

Avoiding password prompts

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

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

# Automating Tasks Using Schedules and Events

## About this chapter

This chapter describes how to use scheduling and event handling features of Sybase IQ to automate database administration and other tasks.

## Contents

Topic	Page
Introduction to scheduling and event handling	740
Understanding schedules	741
Understanding events	742
Understanding event handlers	745
Schedule and event internals	747
Scheduling and event handling tasks	749

## Introduction to scheduling and event handling

Many database administration tasks are best carried out systematically. For example, a regular backup procedure is an important part of proper database administration procedures.

You can automate routine tasks in Sybase IQ by adding an **event** to a database, and providing a **schedule** for the event. Whenever one of the times in the schedule passes, a sequence of actions called an **event handler** is executed by the database server.

Database administration also requires taking action when certain conditions occur. For example, it may be appropriate to e-mail a notification to a system administrator when a disk containing the transaction log is filling up, so that the administrator can handle the situation. These tasks can be automated by defining event handlers for one of a set of **system events**.

### Chapter contents

This chapter contains the following material:

- An introduction to scheduling and event handling (this section).
- Concepts and background information to help you design and use schedules and event handlers:
  - “Understanding schedules” on page 741.
  - “Understanding events” on page 742.
- A discussion of techniques for developing event handlers:
  - “Developing event handlers” on page 746.
- Internals information:
  - “Schedule and event internals” on page 747.
- Step by step instructions for how to carry out automation tasks.
  - “Scheduling and event handling tasks” on page 749.

### Questions and answers

To answer the question...	Consider reading...
What is a schedule?	“Understanding schedules” on page 741
What is a system event?	“Understanding events” on page 742
What is an event handler?	“Understanding event handlers” on page 745
How do I debug event handlers?	“Developing event handlers” on page 746

To answer the question...	Consider reading...
How does the database server use schedules to trigger event handlers?	“How the database server checks for scheduled times” on page 748
How can I schedule regular backups?	For an example, see “Understanding schedules” on page 741
What kind of system events can the database server use to trigger event handlers?	“Understanding events” on page 742
What connection do event handlers get executed on?	“CREATE EVENT statement” in Chapter 6, “SQL Statements” of the <i>Sybase IQ Reference Manual</i>
How do event handlers get information about what triggered them?	“How event handlers are executed” on page 748 “Developing event handlers” on page 746 “EVENT_PARAMETER function [System]” in Chapter 5, “SQL Functions” of the <i>Sybase IQ Reference Manual</i>

## Understanding schedules

By scheduling activities you can ensure that a set of actions is executed at a set of preset times. The scheduling information and the event handler are both stored in the database itself.

You can define complex schedules by associating more than one schedule with a named event.

The following examples give some ideas for scheduled actions that may be useful.

### Examples

- Instruct the database server to carry out an automatic incremental backup, every day at 1 am:

```

CREATE EVENT IncrementalBackup
SCHEDULE
START TIME '1:00AM' EVERY 24 HOURS
HANDLER
BEGIN
    BACKUP DATABASE INCREMENTAL
    TO 'backups/daily.incr'
END

```

- Summarize orders at the end of each business day:

```
CREATE EVENT Summarize
SCHEDULE
START TIME '6:00 pm'
ON ( 'Mon', 'Tue', 'Wed', 'Thu', 'Fri' )
HANDLER
BEGIN
INSERT INTO dba.OrderSummary
    SELECT MAX( date_ordered ),
           COUNT( * ),
           SUM( amount )
    FROM dba.Orders
    WHERE date_ordered = current date
END
```

## Defining schedules

Schedule definitions have several components to them, to permit flexibility:

- **Name** Each schedule definition has a name. You can assign more than one schedule to a particular event, which can be useful in designing complex schedules.
- **Start time** You can define a start time for the event, which is the time that it is first executed.
- **Range** As an alternative to a start time, you can specify a range of times for which the event is active.
- **Recurrence** Each schedule can have a recurrence. The event is triggered on a frequency that can be given in hours, minutes, or seconds, on a set of days that can be specified as days of the week or days of the month.

## Understanding events

The database server tracks several kinds of system events. Event handlers are triggered when the system event is checked by the database server, and satisfies a provided **trigger condition**.

By defining event handlers to execute when a chosen system event occurs and satisfies a trigger condition that you define, you can improve the security and safety of your data, and help to ease administration.

For information on the available system events, see “Choosing a system event.” For information on trigger conditions, see “Defining trigger conditions for events” on page 744.

## Choosing a system event

Sybase IQ tracks several system events. Each system event provides a hook on which you can hang a set of actions. The database server tracks the events for you, and executes the actions (as defined in the event handler) when needed.

The available system events include the following:

- **Backup** You can use the BackupEnd event type to take actions at the end of a backup.
- **DatabaseStart** The database is started.
- **Connection events** When a connection is made (Connect) or when a connection attempt fails (ConnectFailed). You may want to use these events for security purposes.
- **Free disk space** Tracks the available disk space on the device holding the system database file (DBDiskSpace) which is the .db file, the log file (LogDiskSpace), or system temporary file (TempDiskSpace). The disk space event types require are not available on UNIX platforms.

You may want to use disk space events to alert administrators in case of a disk space shortage. Remember, however, that these events do not reflect available disk space for the IQ Store or IQ Temporary Store.

- **File size** The file reaches a specified size. This can be used for the system database file (GrowDB), the transaction log (GrowLog), or the system temporary file space (GrowTemp).

You may want to use file size events to track unusual actions on the database, or monitor bulk operations. Remember, however, that these events do not reflect file sizes for the IQ Store or IQ Temporary Store.

- **SQL errors** When an error is triggered, you can use the RAISERROR event type to take actions.

- **Idle time** The database server has been idle for a specified time. You may want to use this event type to carry out routine maintenance operations at quiet times.

## Defining trigger conditions for events

Each event definition has a system event associated with it. It also has one or more trigger conditions. The event handler is triggered when the trigger conditions for the system event are satisfied.

The trigger conditions are included in the WHERE clause of the CREATE EVENT statement, and can be combined using the AND keyword. Each trigger condition is of the following form:

**event\_condition( condition-name ) comparison-operator value**

The *condition-name* argument is one of a set of preset strings, which are appropriate for different event types. The database server does not check that the condition-name matches the event type: it is your responsibility to ensure that the condition is meaningful in the context of the event type.

---

**Note** The trigger conditions associated with Sybase IQ events are not the same as Adaptive Server Anywhere or Adaptive Server Enterprise triggers, which execute automatically when a user attempts a specified data modification statement on a specified table.

---

### Examples

- Notify an administrator of a possible attempt to break into the database:

```
create event SecurityCheck
type ConnectFailed
handler
begin
declare num_failures int;
declare mins int;

insert into FailedConnections( log_time )
values ( current timestamp );

select count( * ) into num_failures
from FailedConnections
where log_time >= dateadd( minute, -5,
current timestamp );

if( num_failures >= 3 ) then
```

```
select datediff( minute, last_notification,
                 current timestamp ) into mins
from Notification;

if( mins > 30 ) then
    update Notification
    set last_notification = current timestamp;

call xp_sendmail( recipient='DBAdmin',
                  subject='Security Check',
                  "message"=
'over 3 failed connections in last 5 minutes' )
end if
end if
end
```

See the section “Developing event handlers” on page 746 for more information on the use of the system extended stored procedure `xp_sendmail`.

- Run a process when the server has been idle for ten minutes. Do not execute more frequently than once per hour:

```
create event Soak
type ServerIdle
where event_condition( 'IdleTime' ) >= 600
and event_condition( 'Interval' ) >= 3600
handler
begin
message ' Insert your code here ... '
end
```

## Understanding event handlers

Event handlers execute on a separate connection from the action that triggered the event, and so do not interact with client applications. They execute with the permissions of the creator of the event.

## Developing event handlers

Event handlers, whether for scheduled events or for system event handling, contain compound statements, and are similar in many ways to stored procedures. You can add loops, conditional execution, and so on, and you can use the Sybase IQ debugger to debug event handlers.

### Context information for event handlers

One difference between event handlers and stored procedures is that event handlers do not take any arguments. Certain information about the context in which an event was triggered is available through the `EVENT_PARAMETER` function, which supplies information about the connection that caused an event to be triggered (connection ID, user ID), as well as the event name and the number of times it has been executed.

For more information, see “`EVENT_PARAMETER` function [System]” in Chapter 5, “SQL Functions,” in the *Sybase IQ Reference Manual*.

### Actions of event handlers

When a trigger condition is satisfied and an event handler executes, one or more actions are carried out, as defined in the `CREATE EVENT` statement. These actions vary and can include sending an e-mail message, performing a backup, or writing to a file.

Sybase IQ now supports the use of the system extended stored procedures. Microsoft Messaging API (MAPI) stored procedures, such as `xp_sendmail`, are supported only on Windows. The other system extended stored procedures, such as `xp_cmdshell`, `xp_read_file`, and `xp_write_file`, are supported on both Windows and UNIX platforms. On UNIX, an event handler can send an e-mail message by calling `sendmail()` from `xp_cmdshell()`.

For details on the use of the system extended stored procedures, see the chapter “System Procedures and Functions” in the *Adaptive Server Anywhere Reference* manual. Information on how to access this manual is provided in the Release Bulletin for your platform.

### Testing event handlers

During development, you want event handlers to be triggered at convenient times. You can use the `TRIGGER EVENT` statement to explicitly cause an event to execute, even when the trigger condition or scheduled time has not occurred. However, `TRIGGER EVENT` does not cause disabled event handlers to be executed.

For more information, see “`TRIGGER EVENT` statement” in Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

While it is not good practice to develop event handlers on a production database, you can disable event handlers explicitly using the `ALTER EVENT` statement.

	<p>It can be useful to use a single set of actions to handle multiple events. For example, you may want to take a notification action if disk space is limited on any of the devices holding the database or log files. To do this, create a stored procedure and call it in the body of each event handler.</p>
Debugging event handlers	<p>Debugging event handlers is very similar to debugging stored procedures. The event handlers appear in the procedures list.</p> <p>One difference is that, because each event handler runs on its own connection, you must be sure to select All connections before setting a breakpoint in an event handler.</p> <p>For step-by-step instructions, see “Debugging an event handler” on page 750.</p> <p>For an example on using event handling, see “Managing IQ user accounts and connections” on page 556.</p>

## Schedule and event internals

This section describes how the database server processes schedules and event definitions.

### How the database server checks for events

Events are classified according to their **event type**, as specified directly in the CREATE EVENT statement. The event types are of two kinds:

- **Active event types** Some event types are the result of action by the database server itself. These active event types include the start and end of different database actions (BackupEnd and so on) or RAISERROR.

When the database server takes the action, it checks to see whether the trigger conditions defined in the WHERE clause are satisfied, and if so triggers any events defined for that event type.

- **Polled event types** Some event types are not triggered solely by database actions. The free disk space types, as well as the IdleTime types, are of this kind.

For these types of events, the database server polls every thirty seconds, starting approximately thirty seconds after the database server is started.

For the `IdleTime` event type, the database server checks whether the server has been idle for the entire thirty seconds. If no requests have started and none are currently active, it adds the idle check interval time in seconds to the idle time total; otherwise, the idle time total is reset to 0. The value for `IdleTime` is therefore always a multiple of thirty seconds. When `IdleTime` is greater than the interval specified in the trigger condition, event handlers associated with `IdleTime` are fired.

## How the database server checks for scheduled times

The calculation of scheduled event times is done when the database server starts, and each time a scheduled event handler completes.

The calculation of the next scheduled time is based on the increment specified in the schedule definition, with the increment being added to the previous start time. If the event handler takes longer to execute than the specified increment, so that the next time is earlier than the current time, the database server increments until the next scheduled time is in the future.

An event handler that takes sixty-five minutes to execute and is requested to run every hour between 9:00 and 5:00 will run every two hours, at 9:00, 11:00, 1:00, and so on.

If you are running a database server intermittently, and it is not running at a scheduled time, the event handler does not run at startup. Instead, the next scheduled time is computed at startup. If, for example, you schedule a backup to take place every night at one o'clock, but regularly shut down the database server at the end of each work day, the backup never takes place.

## How event handlers are executed

When an event handler is triggered, a temporary internal connection is made, on which the event handler is executed. The handler is *not* executed on the connection that caused the handler to be triggered, and consequently statements such as `MESSAGE ... TO CLIENT`, which interact with the client application, are not meaningful within event handlers.

The temporary connection on which the handler is executed does not count towards the connection limit for licensing purposes.

Event creation requires DBA authority, and events execute with the permissions of their creator. If you wish event handlers to execute with non-DBA authority, you can call a procedure from within the handler, as stored procedures run with the permissions of their creator.

Any event errors are logged to the server console.

## Scheduling and event handling tasks

This section collects together instructions for tasks related to automating tasks with schedules and events.

### Adding a schedule or event to a database

Schedules and events are handled in a similar fashion in SQL.

For background information, see “Understanding schedules” on page 741, and “Understanding events” on page 742.

- ❖ **Adding a schedule or event to a database (SQL)**

- 1 Connect to the database as a user with DBA authority.
- 2 Execute a CREATE EVENT statement.

The CREATE EVENT contains many options, depending on the schedule or event you wish to create. These are explained in detail in other tasks.

For more information, see “CREATE EVENT statement” in Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

### Adding a manually-triggered event to a database

If you create an event handler without a schedule or system event to trigger it, it is executed only when manually triggered.

- ❖ **Adding a manually-triggered event to a database (SQL)**

- 1 Connect to the database as a user with DBA authority.
- 2 Execute a CREATE EVENT statement with no schedule or WHERE clause.  
The restricted syntax of the CREATE EVENT is as follows:

```
CREATE EVENT event-name
HANDLER
BEGIN
... event handler
END
```

If you are developing event handlers, you can add schedules or system events to control the triggering of an event later, using the ALTER EVENT statement.

See also:

- For information on triggering events, see “Triggering an event handler” on page 750.
- For information on altering events, see “ALTER EVENT statement” in Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

## Triggering an event handler

Any event handler can be triggered manually, in addition to those occasions when it executes because of a schedule or system event. Triggering events manually can be useful during development of event handlers, and also, for certain events, in production environments. For example, you may have a monthly sales report scheduled, but from time to time you may want to obtain a sales report for a reason other than the end of the month.

For information on developing event handlers, see “Developing event handlers” on page 746.

### ❖ Triggering an event handler (SQL)

- 1 Connect to the database as a user with DBA authority.
- 2 Execute the TRIGGER EVENT statement, supplying the name of the event.  
For example:

```
TRIGGER EVENT sales_report_event
```

For more information, see “TRIGGER EVENT statement” in Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

## Debugging an event handler

Debugging is a regular part of any software development. Event handlers can be debugged during the development process.

For information on developing event handlers, see “Developing event handlers” on page 746.

For information on using the debugger, see Appendix C, “Debugging Logic in the Database.”.

❖ **Debugging an event handler**

1 From a system command prompt, change directory to the directory holding the Sybase IQ software for your operating system.

2 Enter the following command to start the debugger:

```
dbprdbg
```

3 In the Connect login window, enter user name and password, then click OK.

4 In the Connections window, double click All Connections.

5 In the procedures window, double click the event you wish to debug. The event definition is displayed in the Source window.

6 In the Source window, set a breakpoint.

7 From Interactive SQL or another application, trigger the event handler using the TRIGGER EVENT statement.

8 The execution stops at the breakpoint you have set. You can now use the debugger features to trace execution, local variables, and so on.

## Retrieving information about an event or schedule

Sybase IQ stores information about events, system events, and schedules in the system tables SYSEVENT, SYSEVENTTYPE, and SYSSCHEDULE. When you alter an event using the ALTER EVENT statement, you specify the event name and, optionally, the schedule name. When you trigger an event using the TRIGGER EVENT statement, you specify the event name.

You can list event names by querying the system table SYSEVENT. For example:

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

You can list schedule names by querying the system table SYSSCHEDULE. For example:

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

Each event has a unique event id. Use the event\_id columns of SYSEVENT and SYSSCHEDULE to match the event to the associated schedule.

# XML in the Database

## About this appendix

This appendix uses examples to describe how you can use Java tools to access Extensible Markup Language (XML) documents in Sybase IQ.

Topics in this appendix include an overview of XML, a discussion on using XML in an Sybase IQ database, a simple example for a specific result set, and a more general customizable example for different result sets.

## Contents

Topic	Page
Introduction to XML	753
Using XML in the Sybase IQ database	760
Storing XML data in a Sybase IQ database	763
A customizable example for different result sets	770

## Introduction to XML

Like Hypertext Markup Language (HTML), XML is a markup language and a subset of Standardized General Markup Language (SGML). XML, however, is more complete and disciplined, and allows you to define your own application-oriented markup tags. These properties make XML particularly suitable for data interchange.

You can generate XML-formatted documents from data stored in Sybase IQ and, conversely, store data extracted from XML documents in Sybase IQ. Many of the XML tools needed to generate and process XML documents are written in Java. Java in Sybase IQ provides a good base for XML-SQL applications using both universal and application-specific tools.

This appendix first discusses XML and how to use XML in an Sybase IQ database. It then presents examples to use as guidelines for using XML in your Sybase IQ database.

## **Source code and Javadoc**

The source code for the Java classes described in this appendix is available in `$ASDIR/sample/JavaSql` (UNIX) or `%ASDIR%\sample\JavaSql` (Windows), which also contains Javadoc-generated HTML pages with the specifications of the referenced packages, classes, and methods.

## **Installing XML in Sybase IQ**

A Readme file named *index.html* is also included in the directory `$ASDIR/sample/JavaSql` (UNIX) or `%ASDIR%\sample\JavaSql` (Windows). This file contains information on installing XML and setting up the example Java classes.

## **References**

This appendix presents only an overview of XML. For detailed information, refer to these Web documents:

- World Wide Web Consortium (W3C) at <http://www.w3.org>
- W3C, Document Object Model (DOM) at <http://www.w3.org/DOM/>
- W3C, Extensible Markup Language (XML™) at <http://www.w3.org/XML/>
- W3C, Extensible Stylesheet Language (XSL) at <http://www.w3.org/Style/XSL>
- Sun Microsystems, Inc., Java™ Technology and XML at <http://java.sun.com/xml/index.html>
- SAX Project, Simple API for XML at <http://www.saxproject.org/>

## **An overview of XML**

XML is a markup language and subset of SGML, created to provide functionality that goes beyond that of HTML for Web publishing and distributed document processing.

XML is less complex than SGML, but more complex and flexible than HTML. Although XML and HTML can usually be read by the same browsers and processors, XML has characteristics that make it better able to share documents:

- XML documents possess a strict phrase structure that makes data easy to find and access. For example, opening tags of all elements must have a corresponding closing tag, as in the element `<p>A paragraph.<\p>`.
- XML lets you develop and use tags that distinguish different types of data, for example, customer numbers or item numbers.
- XML lets you create an application-specific document type, which makes it possible to distinguish one kind of document from another.
- XML documents allow different views of the XML data. XML documents contain only markup and content; they do not contain formatting instructions. Formatting instructions are normally provided on the client using Extensible Style Language (XSL) specifications.

## A sample XML document

The sample Order document is designed for a purchase order application. Customers submit orders, which are identified by a date and a customer ID. Each order item has an item ID, an item name, a quantity, and a unit designation.

An order document might display on screen like this:

### **ORDER**

Date: July 4, 1999

Customer ID: 123

Customer Name: Acme Alpha

#### **Items:**

Item ID	Item Name	Quantity
987	Coupler	5
654	Connector	3 dozen
579	Clasp	1

A possible XML representation of the data for the order is:

```
<?xml version="1.0"?>
<Order>
```

```
<Date>1999/07/04</Date>
<CustomerId>123</CustomerId>
<CustomerName>Acme Alpha</CustomerName>
<Item>
    <ItemId> 987</ItemId>
    <ItemName>Coupler</ItemName>
    <Quantity>5</Quantity>
</Item>
<Item>
    <ItemId>654</ItemId>
    <ItemName>Connector</ItemName>
    <Quantity unit="12">3</Quantity>
</Item>
<Item>
    <ItemId>579</ItemId>
    <ItemName>Clasp</ItemName>
    <Quantity>1</Quantity>
</Item>
</Order>
```

The XML document for the order data consists of these parts:

- The *XML declaration*, `<?xml version="1.0"?>`, which identifies Order as an XML document.

XML documents are represented as character data. In each document, the character encoding (character set) is specified, either explicitly or implicitly. To explicitly specify the character set, include the character set in the XML declaration. For example:

```
<?xml version="1.0" encoding="ISO-8859-1">
```

If you do not include the character set in the XML declaration, the default, which is UTF8, is used. For example:

```
<?xml version="1.0"?>
```

---

**Note** When the default character sets of the client and server differ, Sybase IQ bypasses normal character set translations so that the declared character set continues to match the actual character set. See “Character sets and XML data” on page 759.

---

- *User-created element tags* such as `<Order>...</Order>`, `<CustomerId>...</CustomerId>`, `<Item>...</Item>`. In XML documents, all opening tags must have a corresponding closing tag.
- *Text data* such as “Acme Alpha,” “Coupler,” and “579.”

- *Attributes embedded in element tags* such as <Quantity unit = "12">. This kind of coding allows you the flexibility to customize elements.

A document with these parts, and with the element tags strictly nested, is called a **well-formed XML document**. Note that in the example above element tags describe the data they contain, and the document contains no formatting instructions.

## XML document types

A **Document Type Definition** (DTD) defines the structure of a class of XML documents, making it possible to distinguish between classes. A DTD is a list of element and attribute definitions unique to a class. Once you have set up a DTD, you can reference a DTD in another document or embed the DTD in the XML document.

Here is another example of an XML document:

```
<?xml version="1.0"?>
<Info>
  <OneTag>1999/07/04</OneTag>
  <AnotherTag>123</AnotherTag>
  <LastTag>Acme Alpha</LastTag>
  <Thing>
    <ThingId>987</ThingId>
    <ThingName>Coupler</ThingName>
    <Amount>5</Amount>
  </Thing>
  <Thing>
    <ThingId>654</ThingId>
    <ThingName>Connecter</ThingName>
  </Thing>
  <Thing>
    <ThingId>579</ThingId>
    <ThingName>Clasp</ThingName>
    <Amount>1</Amount>
  </Thing>
</Info>
```

This example, called Info, is a well-formed document and has the same structure and data as the XML Order document. However, Info is not recognized by a processor designed for Order documents, because each has a different DTD.

The DTD for XML Order documents is:

```
<!ELEMENT Order (Date, CustomerId, CustomerName,
    Item+)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT CustomerId (#PCDATA)>
<!ELEMENT CustomerName (#PCDATA)>
<!ELEMENT Item (ItemId, ItemName, Quantity)>
<!ELEMENT ItemId (#PCDATA)>
<!ELEMENT ItemName (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ATTLIST Quantity units CDATA #IMPLIED>
```

This DTD specifies that:

- An order consists of these required elements: a date, a customer ID, a customer name, and one or more items indicated by “+”. These items are required. A question mark indicates an optional element, for example, “CustomerName?”. An asterisk indicates that an element can occur zero or more times (for example, “Item\*”).
- Elements defined by “(#PCDATA)” are character text.
- The “<ATTLIST...>” definition specifies that quantity elements have a “units” attribute; the “#IMPLIED” specification indicates that the “units” attribute is optional.

The character text of XML documents is not constrained. For example, there is no way to specify that the text of a quantity element should be numeric, so the following is valid:

```
<Quantity unit="Baker's dozen">three</Quantity>
<Quantity unit="six packs">plenty</Quantity>
```

Restrictions on the text of elements are handled by applications that process XML data.

A DTD follows the `<?xml version="1.0"?>` instruction in an XML document. You can either include the DTD within your XML document, or you can reference an external DTD.

- This example references an external DTD:

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "Order.dtd">
<Order>
...
</Order>
```

- This example contains an embedded DTD:

```
<?xml version="1.0"?>
<!DOCTYPE Order [
<!ELEMENT Order (Date, CustomerId, CustomerName,
```

```
Item+)>
<!ELEMENT Date (#PCDATA)
<!ELEMENT CustomerId (#PCDATA)>
<!ELEMENT CustomerName (#PCDATA)>
<!ELEMENT Item (ItemId, ItemName, Quantity)>
<!ELEMENT ItemId (#PCDATA)>
<!ELEMENT ItemName (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ATTLIST Quantity units CDATA #IMPLIED>
]>
<Order>
<Date>1999/07/04</Date>
<CustomerId>123</CustomerId>
<CustomerName>Acme Alpha</CustomerName>
<Item>
...
</Item>
</Order>
```

A DTD is not required for an XML document. However, a **valid XML document** has a DTD and conforms to that DTD.

## XSL: formatting XML information

You can use XSL to format XML documents. XSL specifications (stylesheets) consist of a set of rules that define the transformation of an XML document into either an HTML document or a different XML document:

- XSL specifications that transform an XML document into HTML can specify normal HTML formatting details in the output HTML.
- XSL specifications that transform an XML document into another XML document can map the input XML document to an output XML document with different element names and phrase structure.

You can create your own stylesheets for the display of particular classes for particular applications. XSL is normally used with presentation applications rather than with applications for data interchange or storage.

## Character sets and XML data

If the declared character sets of your client and server differ, you must be careful when declaring the character set of your XML documents.

Every XML document has a character set value. If that encoding is not declared in the XML declaration, the default value of UTF8 is assumed. The XML processor, when parsing the XML data, reads this value and handles the data accordingly. When the default character set of the client and server differ, Sybase IQ bypasses normal character set conversions to ensure that the declared character set and the actual character set remain the same.

- If you introduce an XML document into the database by providing the complete text in a set variable statement, Sybase IQ translates the entire SQL statement into the server's character set before processing the insertion. This is the way Sybase IQ normally translates character text, and you must ensure that the declared character set of the XML document matches that of the server.
- If you read an XML document from the database, Sybase IQ does *not* translate the character set of the data to that of the client, thus preserving the integrity of the XML document.

## Using XML in the Sybase IQ database

To use XML documents for data interchange in Sybase IQ, you must be able to store in the database the data contained in XML documents and be able to access this data.

### Mapping and storage

Sybase IQ supports **element storage** of XML data. This method extracts data elements from an XML document and stores them as data rows and columns in an Sybase IQ database. All of the data from the XML document is available as normal SQL data that you can query and update using SQL operations.

For example, using the XML Order sample document, you can create SQL tables with columns for the individual elements of an order: *Date*, *CustomerId*, *CustomerName*, *ItemId*, *ItemName*, *Quantity*, and *Units*. You can then manage that data in SQL with normal SQL operations:

- To produce an XML document for Order data contained in SQL, retrieve the data, and assemble an XML document with it.
- To store an XML document with new Order data, extract the elements of that document, and update the SQL tables with that data.

## Client or server considerations

Java methods assemble and disassemble an XML document and reference or update its elements. You can execute these Java methods either on the client or on the server. When you map individual elements of an XML document to SQL data, in most cases, the XML document is larger than the SQL data. It is generally more efficient to assemble and disassemble the XML document on the client and transfer only the SQL data between the client and the server.

## Accessing XML in SQL

This appendix discusses three applications of XML in SQL. These applications are organized in three layers:

- *Transact-SQL statements* such as insert, select, and update for referencing SQL variables that contain XML document data. These SQL operations use Java classes and methods to manipulate the XML documents.
- *Java classes* to contain XML documents and to access and update the elements of these documents. There is an application-specific class for the Order document type and a general class for arbitrary SQL result sets.
- An *XML parser*, which is used by the Java classes to analyze and manipulate XML documents.

The Java classes that are used in this appendix to demonstrate XML applications are JXml, OrderXml, and ResultSetXml.

- JXml stores and parses XML. It does not validate XML documents. It is designed as a base class for subclasses that:
  - Validate specific XML document types
  - Provide application-oriented methods

OrderXml and ResultSetXml are two such subclasses.

- OrderXml illustrates support for an application-specific XML document type. OrderXml validates Order documents for the Order DTD. You can use OrderXml methods to reference and update elements of the Order document.
- ResultSetXml represents SQL result sets. The ResultSetXml constructor validates the ResultSet document for the ResultSet DTD. ResultSetXml methods are used to reference and update elements of the ResultSet document.

ResultSetXml illustrates support for a general XML document type capable of representing arbitrary SQL data.

“The OrderXml class for Order documents” on page 763 and “The ResultSetXml class for result set documents” on page 775 describe these classes and their methods and parameters. For Javadoc HTML pages with detailed specifications for the classes and for source code, refer to `$ASDIR/sample/JavaSql` (UNIX) or `%ASDIR%\sample\JavaSql` (Windows).

## XML parsers

You can analyze XML documents and extract their data using SQL character-string operations such as `substring`, `charindex`, and `patindex`. However, it is more efficient to use Java in SQL and tools written in Java such as XML parsers.

XML parsers perform these functions:

- Check that a document is well-formed and valid.
- Handle character-set issues.
- Generate a Java representation of a document’s parse tree.
- Build or modify a document’s parse tree.
- Generate a document’s text from its parse tree.

Many XML parsers are available with a free license or are in the public domain. The parsers usually implement two standard interfaces: the Simple API for XML (SAX) and the Document Object Model (DOM).

- *SAX* is an interface for parsing. It specifies input sources, character sets, and routines to handle external references. While parsing, it generates events so that user routines can process the document incrementally, and it returns a DOM object that is the parse tree of the document.
- *DOM* is an interface for the parse tree of an XML document. It provides facilities for stepping through and assembling a parse tree.

Applications that use the SAX and DOM interfaces are portable across XML parsers.

## Storing XML data in a Sybase IQ database

This section provides a simple example that demonstrates how you can store the data contained in XML documents in an Sybase IQ database.

The example in this section, the XML Order document type, is designed for a specific purchase-order application, and the Java methods created for it assume a specific set of SQL tables for storing purchase order data.

For a more generalized example, applicable to a range of SQL result sets, see “A customizable example for different result sets” on page 770.

### The *OrderXml* class for Order documents

The example in this section uses the *OrderXml* class and its methods for basic operations on XML Order documents. The source code and Javadoc specifications for *OrderXml* are in \$ASDIR/sample/JavaSql.

*OrderXml* is a subclass of the *JXml* class, which is specialized for XML Order documents. The *OrderXml* constructor validates the document for the Order DTD. Methods of the *OrderXml* class support referencing and updating the elements of the Order document.

- Constructor: *OrderXml(String)*

Validates that the *String* argument contains a valid XML Order document, then constructs an *OrderXml* object containing that document. For example, “*doc*” is a Java string variable containing an XML Order document, perhaps one read from a file:

```
jcs.xml.orders.OrderXml ox = new jcs.xml.orders.OrderXml(doc);
```

- Constructor: *OrderXml(date, customerId, dtdOption, server)*

The parameters are all *String*.

This method assumes a set of SQL tables containing Order data. The method uses JDBC to execute a SQL query that retrieves Order data for the given *date* and *customerId*. The method then assembles an XML Order document with the data.

The *server* parameter identifies the IQ server on which to execute the query.

- If you invoke the method in a client environment, specify the server name.

- If you invoke the method in Sybase IQ (in a SQL statement or in dbisql), specify either an empty string or the string “jdbc:default:connection,” which indicates that the query should be executed on the current IQ server.

The *dtdOption* parameter indicates whether the generated Order contains the DTD or the DTD is referenced externally.

For example:

```
jcs.xml.orders.OrderXml ox = new OrderXml("990704", "123",
    "external", "antibes:4000?user=DBA&password=SQL");
```

- `void order2Sql(String ordersTableName, String server)`

Extracts the elements of the Order document and stores them in a SQL table created by the `createOrdertable()` method. *ordersTableName* is the name of the target table. The *server* parameter is as described for the `OrderXml` constructor. For example, if *ox* is a Java variable of type `OrderXml`:

```
ox.order2Sql("current_orders", "antibes:4000?user=DBA&password=SQL");
```

This call extracts the elements of the Order document contained in *ox*, and uses JDBC to insert the extracted elements into rows and columns of the table named `current_orders`.

- `static void createOrderTable(String ordersTableName, String server)`

Creates a SQL table with columns suitable for storing Order data: `customer_id`, `order_date`, `item_id`, `quantity`, and `unit`. *ordersTableName* is the name of the new table. The *server* parameter is as described for the `OrderXml` constructor. For example:

```
jcs.xml.orders.OrderXml.createOrderTable
    ("current_orders",
     "antibes:4000?user=DBA&password=SQL");
```

- `String getOrderElement(String elementName)`

*elementName* is “Date,” “CustomerId,” or “CustomerName.” The method returns the text of the element. For example, if *ox* is a Java variable of type `OrderXml`:

```
String customerId = ox.getOrderElement("CustomerId");
String customerName = ox.getOrderElement("CustomerName");
String date = ox.getOrderElement("Date");
```

- `void setOrderElement(String elementName, String newValue)`

*elementName* is as described for `getOrderElement()`. The method sets that element to *newValue*. For example, if *ox* is a Java variable of type `OrderXml`:

```
ox.setOrderElement("CustomerName", "Acme Alpha Consolidated");
ox.setOrderElement("CustomerId", "987a");
ox.setOrderElement("Date", "1999/07/05");
```

- `String getItemElement(int itemNumber, String elementName)`

*itemNumber* is the index of an item in the order. *elementName* is “`ItemId`,” “`ItemName`,” or “`Quantity`.” The method returns the text of the item. For example, if *ox* is a Java variable of type `OrderXml`:

```
String itemId = ox.getItemElement(2, "ItemId");
String itemName = ox.getItemElement(2, "ItemName");
String quantity = ox.getItemElement(2, "Quantity");
```

- `void setItemElement(int itemNumber, String elementName, String newValue)`

*itemNumber* and *elementName* are as described for the `getItemElement` method. **setItemElement** sets the element to *newValue*. For example, if *ox* is a Java variable of type `OrderXml`:

```
ox.setItemElement(2, "ItemId", "44");
ox.setItemElement(2, "ItemName", "cord");
ox.setItemElement(2, "Quantity", "3");
```

- `String getItemAttribute(int itemNumber, elementName, attributeName)`

*itemNumber* and *elementName* are described as for `getItemElement`. *elementName* and *attributeName* are both `String`. *attributeName* must be “unit.” The method returns the text of the unit attribute of the item.

---

**Note** Since the Order documents currently have only one attribute, the *attributeName* parameter is unnecessary. It is included to illustrate the general case, for example, if *ox* is a Java variable of type `OrderXml`:

```
String itemId = ox.getItemAttribute(2, "unit");
```

- `void setItemAttribute (int itemNumber, elementName, attributeName, newValue)`

*itemNumber*, *elementName*, and *attributeName* are as described for `getItemAttribute`. *elementName*, *attributeName*, and *newValue* are `String`. The method sets the text of the unit attribute of the item to *newValue*. For example, if *ox* is a Java variable of type `OrderXml`:

```
ox.setItemAttribute(2, "unit", "13");
```

- `void appendItem(newItemId, newItemName, newQuantity, newUnit)`

The parameters are all String. The method appends a new item to the document, with the given element values. For example, if *ox* is a Java variable of type OrderXml:

```
ox.appendItem("77", "spacer", "5", "12");
```

- `void deleteItem(int itemNumber)`

*itemNumber* is the index of an item in the order. The method deletes that item. For example, if *ox* is a Java variable of type OrderXml:

```
ox.deleteItem(2);
```

## Creating and populating SQL tables for Order data

In this section we create several tables. These tables are designed to contain data from XML Order documents, so that we can demonstrate the technique for element data storage.

The following SQL statements create SQL tables customers, orders, and items, whose columns correspond with the elements of the XML Order documents.

```
create table customers
    (customer_id varchar(5) not null unique,
     customer_name varchar(50) not null)

create table orders
    (customer_id varchar(5) not null,
     order_date datetime not null,
     item_id varchar(5) not null,
     quantity int not null,
     unit smallint default 1)

create table items
    (item_id varchar(5) unique,
     item_name varchar(20))
```

We create these tables to accommodate XML Order documents, but they are ordinary IQ tables.

The following SQL statements populate the tables with the data in the example XML Order document (see “A sample XML document” on page 755):

```
insert into customers values('123', 'Acme Alpha')
insert into orders values ('123', '1999/05/07',
                           '987', 5, 1)
```

```

insert into orders values ('123', '1999/05/07',
                          '654', 3, 12)

insert into orders values ('123', '1999/05/07',
                          '579', 1, 1)

insert into items values ('987', 'Widget')

insert into items values ('654',
                         'Medium connecter')

insert into items values ('579',
                         'Type 3 clasp')

```

Use select to retrieve the Order data from the tables:

```

select order_date as Date, c.customer_id as CustomerId,
       customer_name as CustomerName,
       o.item_id as ItemId, i.item_name as ItemName,
       quantity as Quantity, o.unit as unit
  from customers c, orders o, items i
 where c.customer_id=o.customer_id and
       o.item_id=i.item_id

```

Date	Customerid	CustomerName	ItemId	ItemName	Quantity	Unit
July 4 1999	123	Acme Alpha	987	Coupler	5	1
July 4 1999	123	Acme Alpha	654	Connector	3	12
July 4 1999	123	Acme Alpha	579	Clasp	1	1

## Using the element storage technique

This section describes the element storage technique for bridging XML and SQL.

- “Composing Order documents from SQL data” on page 767 discusses the composition of an XML Order document from SQL data.
- “Decomposing data from an XML Order into SQL” on page 769 discusses the decomposition of an XML Order document to SQL data.

## Composing Order documents from SQL data

In this example, Java methods generate an XML Order document from the SQL data in the tables created in “Creating and populating SQL tables for Order data” on page 766.

A constructor method of the OrderXml class maps the data. A call of that constructor is:

```
new jcs.xml.orders.OrderXml("990704", "123",
    "external", "antibes:4000?user=DBA&password=SQL");
```

This constructor method uses internal JDBC operations to:

- Execute a SQL query for the Order data
- Generate an XML Order document with the data
- Return the OrderXml object that contains the Order document

You can invoke the OrderXml constructor on the client or the IQ server.

- If you invoke the OrderXml constructor on the client, the JDBC operations that it performs use jConnect to connect to the IQ server and perform the SQL query. The constructor then reads the result set of that query and generates the Order document on the client.
- If you invoke the OrderXml constructor on the IQ server, the JDBC operations that it performs use the native JDBC driver to connect to the current IQ server and perform the SQL query. The constructor then reads the result set and generates the Order document on the IQ server.

### **Generating an Order on the client**

Designed to be implemented on the client, main( ) invokes the constructor of the OrderXml class to generate an XML Order from the SQL data. That constructor executes a select for the given date and customer id, and assembles an XML Order document from the result.

```
import java.io.*;
import jcs.util.*;
public class Sql2OrderClient {
    public static void main (String args[]) {
        try{
            jcs.xml.orders.Order orders =
                new jcs.xml.orders.OrderXml("990704", "123",
                    "external", "antibes:4000?user=DBA&password=SQL");
            FileUtil.string2File("Order-sql2Order.xml",
                orders.getXmlText());
        } catch (Exception e) {
            System.out.println("Exception:");
            e.printStackTrace();
        }
    }
}
```

## Generating an Order on the server

Designed for the server environment, the following SQL script invokes the constructor of the OrderXml class to generate an XML Order from the SQL data:

```
create variable orders jcs.xml.orders.OrderXml
set orders =
    new jcs.xml.orders.OrderXml('990704', '123',
    'external', '')
```

## Decomposing data from an XML Order into SQL

In this section, you extract elements from an XML Order document and store them in the rows and columns of the SQL Orders tables. The examples illustrate this procedure in both server and client environments.

You decompose the elements using the Java method `order2Sql()` of the `OrderXml` class. Assume that `xmlOrder` is a Java variable of type `OrderXml`:

```
xmlOrder.order2Sql("orders_received", "antibes:4000?user=DBA&password=SQL");
```

The `order2Sql()` call extracts the elements of the XML Order document contained in variable `xmlOrder`, and then uses JDBC operations to insert that data into the SQL table `orders_received`. You can call this method on the client or on Sybase IQ:

- Invoked from the client, `order2Sql()` extracts the elements of the XML Order document on the client, uses jConnect to connect to the IQ server, then uses Transact-SQL insert to place the extracted data into the table.
- Invoked from the server, `order2Sql()` extracts the elements of the XML Order document in the IQ server, uses the native JDBC driver to connect to the current IQ server, then uses Transact-SQL insert to place the extracted data into the table.

## Decomposing the XML document on the client

Invoked from the client, the `main()` method of the `Order2SqlClient` class creates a table named `orders_received` with columns suitable for Order data. `main()` then extracts the elements of the XML Order contained in the file `Order.xml` into rows and columns of `orders_received`. It performs these actions with calls to the static method `OrderXml.createOrderTable()` and the instance method `order2Sql()`.

```
import jcs.util.*;
import jcs.xml.orders.*;
```

```
import java.io.*;
import java.sql.*;
import java.util.*;
public class Order2SqlClient {
    public static void main (String args[]) {
        try{
            String xmlOrder =
                FileUtil.file2String("order.xml");
            OrderXml.createOrderTable("orders_received",
                "antibes:4000?user=DBA&password=SQL");
            xmlOrder.order2Sql("orders_received",
                "antibes:4000?user=DBA&password=SQL");
        } catch (Exception e) {
            System.out.println("Exception:");
            e.printStackTrace();
        }
    }
}
```

#### Decomposing the XML document on the server

Invoked from the server, the following SQL script invokes the OrderXml constructor to generate an XML Order document from the SQL tables, and then invokes the method OX.sql2Order( ), which extracts the Order data from the generated XML and inserts it into the orders\_received table.

```
create variable xmlorder jcs.xml.orders.OrderXml
set xmlorder = new jcs.xml.orders.OrderXml('19990704',
    '123',
    'external', '')
select xmlorder>>order2Sql('orders_received', '')
```

## A customizable example for different result sets

This section demonstrates how you can store the data from XML documents in a Sybase IQ database using the ResultSet class and its methods for handling result sets. You can customize the ResultSet class for your database application.

Contrast the ResultSet document type and the Order document type:

- The Order document type is a simplified example designed for a specific purchase-order application, and its Java methods are designed for a specific set of SQL tables for purchase order data. See “Storing XML data in a Sybase IQ database” on page 763.
- The ResultSet document type is designed to accommodate many kinds of SQL result sets, and the Java methods designed for it include parameters to accommodate different kinds of SQL queries.

For this example, you create and work with XML ResultSet documents that contain the same data as the XML Order documents.

First, create the orders table and its data:

```
create table orders
  (customer_id varchar(5) not null,
  order_date datetime not null,
  item_id varchar(5) not null,
  quantity int not null,
  unit smallint default 1)
insert into orders values ('123', '1999/05/07', '987', 5, 1)
insert into orders values ('123', '1999/05/07', '654', 3, 12)
insert into orders values ('123', '1999/05/07', '579', 1, 1)
```

## The ResultSet document type

ResultSet documents consist of ResultSetMetaData followed by ResultSetData as shown in the following general form:

```
<?xml version="1.0"?>
<!DOCTYPE ResultSet SYSTEM 'ResultSet.dtd'>
<ResultSet>
  <ResultSetMetaData>
    ...
  </ResultSetMetaData>
  <ResultSetData>
    ...
  </ResultSetData>
</ResultSet>
```

The ResultSetMetaData portion of an XML ResultSet consists of the SQL metadata returned by the methods of the JDBC ResultSet class. The ResultSetMetaData for the example result set is:

```
<ResultSetMetaData
  getColumnCount="7" >
```

```
<ColumnMetaData  
    getColumnDisplaySize="25"  
    getColumnLabel="Date"  
    getColumnName="Date"  
    getColumnType="93"  
    getPrecision="0"  
    getScale="0"  
    isAutoIncrement="false"  
    isCurrency="false"  
    isDefinitelyWritable="false"  
    isNullable="false"  
    isSigned="false" />  
<ColumnMetaData  
    getColumnDisplaySize="5"  
    getColumnLabel="CustomerId"  
    getColumnName="CustomerId"  
    getColumnType="12"  
    getPrecision="0"  
    getScale="0"  
    isAutoIncrement="false"  
    isCurrency="false"  
    isDefinitelyWritable="false"  
    isNullable="false"  
    isSigned="false" />  
<ColumnMetaData  
    getColumnDisplaySize="50"  
    getColumnLabel="CustomerName"  
    getColumnName="CustomerName"  
    getColumnType="12"  
    getPrecision="0"  
    getScale="0"  
    isAutoIncrement="false"  
    isCurrency="false"  
    isDefinitelyWritable="false"  
    isNullable="false"  
    isSigned="false" />  
<ColumnMetaData  
    getColumnDisplaySize="5"  
    getColumnLabel="ItemId"  
    getColumnName="ItemId"  
    getColumnType="12"  
    getPrecision="0"  
    getScale="0"  
    isAutoIncrement="false"  
    isCurrency="false"  
    isDefinitelyWritable="false"
```

```
        isNullable="false"
        isSigned="false" />
<ColumnMetaData
        getColumnDisplaySize="20"
        getColumnLabel="ItemName"
        getColumnName="ItemName"
        getColumnType="12"
        getPrecision="0"
        getScale="0"
        isAutoIncrement="false"
        isCurrency="false"
        isDefinitelyWritable="false"
        isNullable="false"
        isSigned="false" />
<ColumnMetaData
        getColumnDisplaySize="11"
        getColumnLabel="Quantity"
        getColumnName="Quantity"
        getColumnType="4"
        getPrecision="0"
        getScale="0"
        isAutoIncrement="false"
        isCurrency="false"
        isDefinitelyWritable="false"
        isNullable="false"
        isSigned="true" />
<ColumnMetaData
        getColumnDisplaySize="6"
        getColumnLabel="unit"
        getColumnName="unit"
        getColumnType="5"
        getPrecision="0"
        getScale="0"
        isAutoIncrement="false"
        isCurrency="false"
        isDefinitelyWritable="false"
        isNullable="false"
        isSigned="true" />
</ResultSetMetaData>
```

The names of the attributes of ColumnMetaData are simply the names of the methods of the JDBC ResultSetMetaData class, and the values of those attributes are the values returned by those methods.

The ResultSetData portion of an XML ResultSet document is a list of *Row* elements, each having a list of *Column* elements. The text value of a *Column* element is the value returned by the JDBC getString( ) method for the column. The ResultSetData for the example is:

```
<ResultSetData>
  <Row>
    <Column name="Date">1999-07-04 00:00:00.0</Column>
    <Column name="CustomerId">123</Column>
    <Column name="CustomerName">Acme Alpha</Column>
    <Column name="ItemId">987</Column>
    <Column name="ItemName">Coupler</Column>
    <Column name="Quantity">5</Column>
    <Column name="unit">1</Column>
  </Row>
  <Row>
    <Column name="Date">1999-07-04 00:00:00.0</Column>
    <Column name="CustomerId">123</Column>
    <Column name="CustomerName">Acme Alpha</Column>
    <Column name="ItemId">654</Column>
    <Column name="ItemName">Connecter</Column>
    <Column name="Quantity">3</Column>
    <Column name="unit">12</Column>
  </Row>
  <Row>
    <Column name="Date">1999-07-04 00:00:00.0</Column>
    <Column name="CustomerId">123</Column>
    <Column name="CustomerName">Acme Alpha</Column>
    <Column name="ItemId">579</Column>
    <Column name="ItemName">Clasp</Column>
    <Column name="Quantity">1</Column>
    <Column name="unit">1</Column>
  </Row>
</ResultSetData>
</ResultSet>
```

## The XML DTD for the ResultSetXml document type

The DTD for the XML ResultSet document type is:

```
<!ELEMENT ResultSet (ResultSetMetaData,
  ResultSetData)>
<!ELEMENT ResultSetMetaData (ColumnMetaData)+>
  <!ATTLIST ResultSetMetaData getColumnCount CDATA
    #IMPLIED>
<!ELEMENT ColumnMetaData EMPTY>
<!ATTLIST ColumnMetaData
```

```
getCatalogName CDATA #IMPLIED
getColumnDisplaySize CDATA #IMPLIED
getColumnLabel CDATA #IMPLIED
getColumnName CDATA #IMPLIED
getColumnType CDATA #REQUIRED
getColumnTypeName CDATA #IMPLIED
getPrecision CDATA #IMPLIED
getScale CDATA #IMPLIED
getSchemaName CDATA #IMPLIED
getTablelename CDATA #IMPLIED
isAutoIncrement (true|false) #IMPLIED
isCaseSensitive (true|false) #IMPLIED
isCurrency (true|false) #IMPLIED
isDefinitelyWritable (true|false) #IMPLIED
isNullable (true|false) #IMPLIED
isReadOnly (true|false) #IMPLIED
isSearchable (true|false) #IMPLIED
isSigned (true|false) #IMPLIED
isWritable (true|false) #IMPLIED
>
<!ELEMENT ResultSetData (Row)*>
<!ELEMENT Row (Column)+>
<!ELEMENT Column (#PCDATA)>
<!ATTLIST Column
    null (true | false) "false"
    name CDATA #IMPLIED
```

## The *ResultSetXml* class for result set documents

This section describes the *ResultSetXml* class that supports the *ResultSet* DTD.

The *ResultSetXml* class is similar to the *OrderXml* class. It is a subclass of the *JXml* class, which validates a document with the XML *ResultSet* DTD, and also provides methods for accessing and updating the elements of the contained XML *ResultSet* document.

- Constructor: *ResultSetXml(String)*

Validates that the argument contains a valid XML *ResultSet* document and constructs a *ResultSetXml* object containing that document. For example, if *doc* is a Java String variable containing an XML *ResultSet* document, read from a file:

```
jcs.xml.resultset.ResultSetXml rsx =
    new jcs.xml.resultset.ResultSetXml(doc);
```

- Constructor: *ResultSetXml(query, cdataColumns, colNames, dtdOption, server)*

The parameters are all String.

The query parameter is any SQL query that returns a result set.

The server parameter identifies the IQ server on which to execute the query.

- If you invoke the method in a client environment, specify the server name.
- If you invoke the method on an IQ server (in a SQL statement or dbisql), specify either an empty string or the string “jdbc:default:connection,” indicating that the query should be executed on the current Sybase IQ.

The method connects to the server, executes the query, retrieves the SQL result set, and constructs a *ResultSetXml* object with that result set.

The *cdataColumns* parameter indicates which columns should be XML CDATA sections. The *colNames* parameter indicates whether the resulting XML should specify “name” attributes in the “Column” elements. The *dtdOption* indicates whether the resulting XML should include the XML DTD for the ResultSet document type in-line, or reference it externally.

For example:

```
jcs.xml.resultset.ResultSetXml rsx =  
    new jcs.xml.resultset.ResultSetXml  
    ("select 1 as 'a', 2 as 'b', 3 ", "none", "yes",  
     "external", "antibes:4000?user=DBA&password=SQL");
```

This constructor call connects to the server specified in the last argument, evaluates the SQL query given in the first argument, and returns an XML ResultSet containing the data from the result set of the query. This simple SQL query does not reference a table. If the constructor is called in the Sybase IQ, then the server parameter should be an empty string or *jdbc:default:connection*, to indicate a connection to the current server.

- *String toSqlScript(resultTableName, columnPrefix, writeOption, goOption)*

The parameters are all String.

The method returns a SQL script with a create statement and a list of insert statements that re-create the result set data.

The *resultTableName* parameter is the table name for the create and insert statements. (SQL result sets do not specify a table name because they could be derived from joins or unions.) The *columnPrefix* parameter is the prefix to use in generated column names, which are needed for unnamed columns in the result set. The *writeOption* parameter indicates whether the script is to include the create statement, the insert statements, or both. The *goOption* parameter indicates whether the script is to include the go commands, which are required in dbisql and not supported in JDBC.

For example, if *rsx* is a Java variable of type ResultSetXml:

```
rsx>>toSqlScript('systypes_copy', 'column_', 'both', 'yes')
```

- `String getColumn(int rowNumber, int columnNumber)`

*rowNumber* is the index of a row in the result set; *columnNumber* is the index of a column of the result set. The method returns the text of the specified column.

For example, if *rsx* is a Java variable of type ResultSetXml:

```
select rsx>>getColumn(3, 4)
```

- `String getColumn(int rowNumber, String columnName)`

*rowNumber* is the index of a row in the result set; *columnName* is the name of a column of the result set. The method returns the text of the specified column.

For example, if *rsx* is a Java variable of type ResultSetXml:

```
select rsx>>getColumn(3, 'name')
```

- `void setColumn(int rowNumber, int columnNumber, newValue)`

*rowNumber* and *columnNumber* are as described for `getColumn()`. The method sets the text of the specified column to *newValue*.

For example, if *rsx* is a Java variable of type ResultSetXml:

```
set rsx = rsx>>setColumn(3, 4, 'new value')
```

- `void setColumn(int rowNumber, String columnName, newValue)`

*rowNumber* and *columnName* are as described for `getColumn()`. The method sets the text of the specified column to *newValue*.

For example, if *rsx* is a Java variable of type ResultSetXml:

```
set rsx = rsx>>setColumn(3, 'name', 'new value')
```

- `Boolean allString(int columnNumber, String compOp, String comparand)`

*columnNumber* is the index of a column of the result set. *compOp* is a SQL comparison operator (*<*, *>*, *=*, *!=*, *<=*, *>=*). *comparand* is a comparison value. The method returns a value indicating whether the specified comparison is true for all rows of the result set.

For example, if *rsx* is a Java variable of type *ResultSetXml*:

```
if rsx>>allString(3, '<', 'compare value')...
```

This condition is true if, in the result set represented by *rsx*, the value of column 3 is less than “compare value” for all rows. This is a *String* comparison. Similar methods can be used for other data types.

- Boolean *someString*(int *columnNumber*, String *compOp*, String *comparand*)  
*columnNumber* is the index of a column of the result set. *compOp* is a SQL comparison operator (*<*, *>*, *=*, *!=*, *<=*, *>=*). *comparand* is a comparison value. The method returns a value indicating whether the specified comparison is true for some row of the result set.

For example, if *rsx* is a Java variable of type *ResultSetXml*:

```
if rsx>>someString(3, '<', 'compare value') ...
```

This condition is true if, in the result set represented by *rsx*, the value of column 3 is less than “compare value” for some row.

## Using the element storage technique

This section uses the *orders* table to illustrate mapping between SQL data and XML *ResultSet* documents.

- In “Composing a *ResultSet* XML document from the SQL data” on page 778, we generate an XML *ResultSet* document from the SQL data. We assume that we are the *originator* of the XML *ResultSet* document. We use the resulting XML *ResultSet* document to describe the *ResultSet* DTD.
- In “Decomposing the XML *ResultSet* to SQL data” on page 780, we re-generate SQL data from the XML *ResultSet* document. We assume we are the *recipient* of the XML *ResultSet* document.

## Composing a *ResultSet* XML document from the SQL data

You can use Java methods to evaluate a given query and generate an XML result set with the query’s data. This example uses a constructor method of the *ResultSetXml* class. For example:

```

new jcs.xml.resultset.ResultSetXml
("select 1 as 'a', 2 as 'b', 3 ", "none",
"yes", "external",
"antibes:4000?user=DBA&password=SQL") ;

```

The method uses internal JDBC operations to execute the argument query, and then constructs the XML ResultSet for the query's data.

We can invoke this constructor in a client or in the Sybase IQ:

- If you invoke the constructor on a client, specify a server parameter that identifies the IQ server to use when evaluating the query. The query is evaluated on Sybase IQ, but the XML document is assembled on the client.
- If you invoke the constructor on Sybase IQ, specify a null value or *jdbc:default:connection* for the server. The query is evaluated on the current server and the XML document is assembled there.

### Generating a ResultSet on the client

The main( ) method of the OrderResultSetClient class is invoked in a client environment. main( ) invokes the constructor of the ResultSetXml class to generate an XML ResultSet. The constructor executes the query, retrieves its metadata and data using JDBC ResultSet methods, and assembles an XML ResultSet document with the data.

```

import java.io.*;
import jcs.util.*;
public class OrderResultSetClient {
    public static void main (String[] args) {
        try{
            String orderQuery = "select order_date as Date,
                c.customer_id as CustomerId,
                "
                + "customer_name as CustomerName,
                "
                + "o.item_id as ItemId, i.item_name as ItemName,
                "
                + "quantity as Quantity, o.unit as unit "
                + "from customers c, orders o, items i "
                + "where c.customer_id=o.customer_id and
                o.item_id=i.item_id " ;
            jcs.xml.resultset.ResultSetXml rsx
                = new jcs.xml.resultset.ResultSetXml(orderQuery,
                "none", "yes", "external",
                "antibes:4000?user=DBA&password=SQL") ;
            FileUtil.string2File("OrderResultSet.xml",
                rsx.toString());
        } catch (Exception e) {

```

```
        System.out.println("Exception:");
        e.printStackTrace();
    }
}
}
```

## Generating a ResultSet in Sybase IQ

The following SQL script invokes the constructor of the `ResultSetXml` class in a server environment:

```
create variable rsx jcs.xml.resultset.ResultSetXml;
set rsx = new jcs.xml.resultset.ResultSetXml
('select 1 as ''a'', 2 as ''b'', 3 ', 'none', 'yes', 'external', '');
```

## Decomposing the XML ResultSet to SQL data

In this section, you decompose an existing `ResultSet` document to SQL data.

In the section “Decomposing data from an XML Order into SQL” on page 769, you invoke the `order2Sql()` method of the `OrderXml` class to decompose an XML Order document into SQL data. `order2Sql()` directly inserts the extracted data into a SQL table.

In this example, the `toSqlScript()` method of the `ResultSetXml` class decomposes an XML ResultSet document into SQL data. Instead of directly inserting extracted data into a SQL table, however, `toSqlScript()` returns a SQL script with generated insert statements.

The two approaches are equivalent.

## Decomposing the XML ResultSet document on the client

The `main()` method of `ResultSetXml` is executed in a client environment. It copies the file `OrderResultSet.xml`, constructs a `ResultSetXml` object containing the contents of that file, and invokes the `toSqlScript()` method of that object to generate a SQL script that recreates the data of the result set. The method stores the SQL script in the file `order-resultset-copy.sql`.

```
import java.io.*;
import jcs.util.*;
public class ResultSet2Sql{
    public static void main (String[] args) {
        try{
            String xml = FileUtil.file2String ("OrderResultSet.xml");
            jcs.xml.resultset.ResultSetXml rsx
```

```
        = new jcs.xml.resultset.ResultSetXml(xml);
    String sqlScript
        = rsx.toSqlScript("orderresultset_copy", "col_",
                           "both", "no");
    FileUtil.string2File("order-resultset-copy.sql",
                         sqlScript);
    jcs.util.ExecSql.statement(sqlScript,
                               "antibes:4000?user=DBA&password=SQL");
} catch (Exception e) {
    System.out.println("Exception:");
    e.printStackTrace();
}
}
```

The following is the SQL script generated by ResultSet2Sql.

```
set quoted_identifier on
create table orderresultset_copy (
    Date datetime not null ,
    CustomerId varchar (5) not null ,
    CustomerName varchar (50) not null ,
    ItemId varchar (5) not null ,
    ItemName varchar (20) not null ,
    Quantity integer not null ,
    unit smallint not null
)
insert into orderresultset_copy values (
    '1999-07-04 00:00:00.0',    '123',
    'Acme Alpha',      '987', 'Widget', 5,  1 )
insert into orderresultset_copy values (
    '1999-07-04 00:00:00.0',    '123',
    'Acme Alpha',      '654',
    'Medium connecter', 3,  12 )
insert into orderresultset_copy values (
    '1999-07-04 00:00:00.0',    '123',
    'Acme Alpha',      '579', 'Type 3 clasps', 1,  1 )
```

The SQL script includes the set quoted\_identifier on command for those cases in which the generated SQL uses quoted identifiers.

## Decomposing the XML ResultSet document in Sybase IQ

The following SQL script invokes the `toSqlScript( )` method in Sybase IQ and then creates and populates a table with a copy of the result set data.

```
create variable rsx jcs.xml.resultset.ResultSetXml;
set rsx = new jcs.xml.resultset.ResultSetXml
```

```
('select 1 as ''a'', 2 as ''b'', 3 ', 'none', 'yes',
  'external', '');
create variable script java.lang.String;
set script = ( select
  rsx>>toSqlScript('resultset_copy', 'column_', 'both',
  'no'));
select jcs.util.ExecSql.statement(script, '');
```

## XML ResultSet documents: invalid XML characters

This section describes two techniques for dealing with XML markup characters in the result set.

- When data values contain XML markup characters, you can enclose these values in a CDATA section.
- When column names are quoted identifiers that contain XML markup characters, you can substitute the quotes and markup characters with CML entity symbols.

### Using CDATA sections

The *cdata* parameter of the `ResultSetXml` constructor indicates which (if any) columns of the SQL result set contain character data to be bracketed as CDATA sections in the output XML. The *cdata* parameter can be “all,” “none,” or a string of zero or one characters, where a “1” in the *n*<sup>th</sup> position indicates that the *n*<sup>th</sup> column should be bracketed as a CDATA section.

For example, create the table `cdata` in which data values in columns 2, 3, and 4 contain XML markup characters that must be bracketed as CDATA section in the output:

```
create table cdata (
  id int,
  a varchar(250),
  b varchar(250),
  c varchar(250)
)
go
insert into cdata values (
  1,
  '<p>some samples:</p><ol><li>first</li>
   <li>second</li></ol>',
  'x > y || w & z',
  'x > y || w & z'
```

```
)
```

The following SQL statement generates an XML ResultSet document for this table, specifying a value “0111” for the *cdata* parameter.

```
create variable rsx jcs.xml.resultset.ResultSetXml
set rsx = new jcs.xml.resultset.ResultSetXml
("select * from cdata", '0111', 'yes', 'external', '')
```

The following SQL statement generates a SQL script for that XML ResultSet:

```
create variable script java.lang.String
go
set script =
(select rsx>>toSqlScript('markup_col_names',
'col_', 'both', 'yes'))
```

The following select statement returns the contents of the XML ResultSet:

```
select rsx>>toString()
```

This is the XML ResultSet:

```
<?xml version="1.0"?>
<!DOCTYPE ResultSet SYSTEM 'ResultSet.dtd'>
<ResultSet>
    <ResultSetMetaData getColumnCount="4">
        <ColumnMetaData getColumnDisplaySize="11" getColumnLabel="id"
getColumnName="id" getColumnType="4" getPrecision="0" getScale="0"
isAutoIncrement="false" isCurrency="false" isDefinitelyWritable="false"
isNullable="false" isSigned="true" />
        <ColumnMetaData getColumnDisplaySize="250" getColumnLabel="a"
getColumnName="a" getColumnType="12" getPrecision="0" getScale="0"
isAutoIncrement="false" isCurrency="false" isDefinitelyWritable="false"
isNullable="false" isSigned="false" />
        <ColumnMetaData getColumnDisplaySize="250" getColumnLabel="b"
getColumnName="b" getColumnType="12" getPrecision="0" getScale="0"
isAutoIncrement="false" isCurrency="false" isDefinitelyWritable="false"
isNullable="false" isSigned="false" />
        <ColumnMetaData getColumnDisplaySize="250" getColumnLabel="c"
getColumnName="c" getColumnType="12" getPrecision="0" getScale="0"
isAutoIncrement="false" isCurrency="false" isDefinitelyWritable="false"
isNullable="false" isSigned="false" />
    </ResultSetMetaData>
    <ResultSetData>
        <Row>
            <Column name="id">1</Column>
            <Column name="a">
                <! [CDATA[<p>some samples:
                    </p><ol><li>first</li><li>second</li></ol>]]>
```

```
</Column>
<Column name="b">
    <! [CDATA[x > y || w & z]]>
</Column>
<Column name="c">
    <! [CDATA[x > y || w & z]]>
</Column>
</Row>
</ResultSetData>
</ResultSet>
```

This is the SQL script:

```
set quoted_identifier on
create table markup_col_names (
    id integer not null ,
    a varchar (250) not null ,
    b varchar (250) not null ,
    c varchar (250) not null
)
insert into markup_col_names values (
    1,
    '<p>some
samples:</p><ol><li>first</li><li>second</li></ol>',
    'x > y || w & z',
    'x > y || w & z'
)
```

## Column names

The XML generated for a SQL result set specifies the column names of the result set in the ResultSetMetaData section and in the ResultSetData section.

The following SQL select statement specifies a result set:

```
select 1 as 'A>2', 2 as 'B & 3', 3 as 'A<<b', 4 as
'D ''or'' e'
```

The result set has a single row, whose column values are 1, 2, 3, and 4. The column names of those columns are quoted identifiers that contain XML markup characters.

Since the ResultSetXml document for such a result set specifies the column names in XML attributes, the quotation marks and XML markup characters in these names must be replaced with XML entity symbols.

This problem cannot be handled with CDATA sections, since you cannot use CDATA sections in attribute values.

The following example illustrates how to replace the quotation marks and XML markup characters in the column names with XML entity symbols. A SQL script generates the ResultSetXml document for the result set, then generates the SQL script for that ResultSetXml document.

First, SQL statements generate the XML ResultSet document and store it in the variable *rsx*:

```
create variable rsx jcs.xml.resultsets.ResultSetXml
set rsx = new jcs.xml.resultsets.ResultSetXml(
'select 1 as ''A > 2'', 2 as ''b & 3'',
3 as ''a<<b'', 4 as ''d '''or''' e'''',
'none', 'yes', 'external', '' )
```

Then SQL statements generate the SQL script for the XML ResultSet:

```
create variable script java.lang.String
set script = rsx>>toSqlScript('markup_col_names',
'col_',
'create', 'yes')
```

The following select statement returns the contents of the XML document:

```
select rsx>>toString()
```

The XML ResultSet document for this example is:

```
<?xml version="1.0"?>
<!DOCTYPE ResultSet SYSTEM 'ResultSet.dtd'>
<ResultSet>
  <ResultSetMetaData getColumnCount="4">
    <ColumnMetaData getColumnDisplaySize="11"
      getColumnLabel="A > 2" getColumnName="A > 2"
      getColumnType="4" getPrecision="0" getScale="0"
      isAutoIncrement="false" isCurrency="false"
      isDefinitelyWritable="false" isNullable="false"
      isSigned="true" />
    <ColumnMetaData getColumnDisplaySize="11"
      getColumnLabel="b & 3" getColumnName="b & 3"
      getColumnType="4" getPrecision="0" getScale="0"
      isAutoIncrement="false" isCurrency="false"
      isDefinitelyWritable="false" isNullable="false"
      isSigned="true" />
    <ColumnMetaData getColumnDisplaySize="11"
      getColumnLabel="a<<b" getColumnName="a<<b"
      getColumnType="4" getPrecision="0" getScale="0"
      isAutoIncrement="false" isCurrency="false"
      isDefinitelyWritable="false" isNullable="false"
      isSigned="true" />
```

```
<ColumnMetaData getColumnDisplaySize="11"
getColumnName="d 'or' e" getColumnLabel="d 'or' e"
getColumnType="4" getPrecision="0" getScale="0"
isAutoIncrement="false" isCurrency="false"
isDefinitelyWritable="false" isNullable="false"
isSigned="true" />
</ResultSetMetaData>
<ResultSetData>
<Row>
<Column name="A > 2">1</Column>
<Column name="b & 3">2</Column>
<Column name="a<<b">3</Column>
<Column name="d 'or' e">4</Column>
</Row>
</ResultSetData>
</ResultSet>
```

The following is the output SQL script for this example:

```
set quoted_identifier on
create table markup_col_names (
'A > 2' integer not null ,
'b & 3' integer not null ,
'a<<b' integer not null ,
'd ''or'' e' integer not null
```

# Data Access Using JDBC

## About this appendix

This appendix describes how to use JDBC to access data.

JDBC can be used both from client applications and inside the database. Java classes using JDBC provide a more powerful alternative to SQL stored procedures for incorporating programming logic in the database.

## Contents

Topic	Page
JDBC overview	787
Establishing JDBC connections	792
Using JDBC to access data	800
Using the Sybase jConnect JDBC driver	808
Creating distributed applications	813

## JDBC overview

JDBC provides a SQL interface for Java applications: if you want to access relational data from Java, you do so using JDBC calls.

Rather than a thorough guide to the JDBC database interface, this appendix provides some simple examples to introduce JDBC and illustrates how you can use it inside and outside the server. As well, this appendix provides more details on the server-side use of JDBC, running inside the database server.

The examples illustrate the distinctive features of using JDBC in Sybase IQ. For more information about JDBC programming, see any JDBC programming book.

## JDBC and Sybase IQ

You can use JDBC with Sybase IQ in the following ways:

- JDBC on the client** Java client applications can make JDBC calls to Sybase IQ. The connection takes place through the Sybase jConnect JDBC driver or through the iAnywhere JDBC driver.

In this appendix, the phrase **client application** applies both to applications running on a user's machine and to logic running on a middle-tier application server.

- **JDBC in the server** Java classes installed into a database can make JDBC calls to access and modify data in the database, using an internal JDBC driver.

The focus in this appendix is on server-side JDBC.

#### JDBC resources

- **Required software** You need TCP/IP to use the Sybase jConnect driver.

The Sybase jConnect driver may already be available, depending on your installation of Sybase IQ.

For more information about the jConnect driver and its location, see “The jConnect driver files” on page 809.

## Choosing a JDBC driver

Two JDBC drivers are provided for Sybase IQ:

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

For jConnect documentation, see <http://sybooks.sybase.com/jc.html> at <http://sybooks.sybase.com/jc.html>.

- **iAnywhere JDBC driver** This driver communicates with Sybase IQ using the Command Sequence client/server protocol. Its behavior is consistent with ODBC, embedded SQL, and OLE DB applications.

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

- **Features** Both drivers are JDK 2 compliant. The iAnywhere JDBC driver provides fully-scrollable cursors, which are not available in jConnect.
- **Pure Java** The jConnect driver is a pure Java solution. The iAnywhere JDBC driver requires the Sybase IQ or Adaptive Server Anywhere ODBC driver and is not a pure Java solution.
- **Performance** The iAnywhere JDBC driver provides better performance for most purposes than the jConnect driver.

- **Compatibility** The TDS protocol used by the jConnect driver is shared with Adaptive Server Enterprise. Some aspects of the driver's behavior are governed by this protocol, and are configured to be compatible with Adaptive Server Enterprise.

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

#### JDBC considerations

Consider the following when running Java applications:

- An issue exists when connecting to a Sybase IQ 12.5 server through DBISQL Java using the iAnywhere JDBC driver. For details, see “Data truncation or data conversion error” in *Sybase IQ Troubleshooting and Recovery Guide*.
- Java applications running in Sybase IQ run slower than when run outside in a Sun Java Virtual Machine (JVM). Despite this limitation, Sybase recommends that you tune your applications by increasing the available memory for IQ JVM use with the database options JAVA\_HEAP\_SIZE and JAVA\_NAMESPACE\_SIZE. See Chapter 2, “Database Options,” in the *Sybase IQ Reference Manual*.

## JDBC program structure

The following sequence of events typically occur in JDBC applications:

- 1 **Create a Connection object** Calling a getConnection class method of the DriverManager class creates a Connection object, and establishes a connection with a database.
- 2 **Generate a Statement object** The Connection object generates a Statement object.
- 3 **Pass a SQL statement** A SQL statement that executed within the database environment passes to the Statement object. If the statement is a query, this action returns a ResultSet object.

The ResultSet object contains the data returned from the SQL statement, but exposes it one row at a time (similar to the way a cursor works).

- 4 **Loop over the rows of the result set** The next method of the ResultSet object performs two actions:
  - The current row (the row in the result set exposed through the ResultSet object) advances one row.

- A Boolean value (true/false) returns to indicate whether there is, in fact, a row to advance to.

**5 For each row, retrieve the values** Values are retrieved for each column in the ResultSet object by identifying either the name or position of the column. You can use the getDate method to get the value from a column on the current row.

Java objects can use JDBC objects to interact with a database and get data for their own use, for example to manipulate or for use in other queries.

## Server-side JDBC features

JDBC 1.2 is part of JDK 1.1. JDBC 2.0 is part of Java 2 (JDK 1.2).

Java in the database supplies a subset of the JDK version 1.1, so the internal JDBC driver supports JDBC version 1.2.

The internal JDBC driver (`asajdbc`) makes some features of JDBC 2.0 available from server-side Java applications, but does not provide full JDBC 2.0 support.

The JDBC classes in the `java.sql` package that is part of the Java in the database support are at level 1.2. Server-side features that are part of JDBC 2.0 are implemented in the `sybase.sql.ASA` package. To use JDBC 2.0 features you must cast your JDBC objects into the corresponding classes in the `sybase.sql.ASA` package, rather than the `java.sql` package. Classes that are declared as `java.sql` are restricted to JDBC 1.2 functionality only.

The classes in `sybase.sql.ASA` are as follows:

JDBC class	Sybase internal driver class
<code>java.sql.Connection</code>	<code>sybase.sql.ASA.SAConnection</code>
<code>java.sql.Statement</code>	<code>sybase.sql.ASA.SAStatement</code>
<code>java.sql.PreparedStatement</code>	<code>sybase.sql.ASA.SAPreparedStatement</code>
<code>java.sql.CallableStatement</code>	<code>sybase.sql.ASA.SACallableStatement</code>
<code>java.sql.ResultSetMetaData</code>	<code>sybase.sql.ASA.SAResultSetMetaData</code>
<code>java.sql.ResultSet</code>	<code>sybase.sql.SAResultSet</code>
<code>java.sql.DatabaseMetaData</code>	<code>sybase.sql.SADatabaseMetaData</code>

The following function provides a `ResultSetMetaData` object for a prepared statement without requiring a `ResultSet` or executing the statement. This function is not part of the JDBC standard.

`ResultSetMetaData`

```
sybase.sql.ASA.SAPreparedStatement.describe()
```

**JDBC 2.0 restrictions**

The following classes are part of the JDBC 2.0 core interface, but are not available in the `sybase.sql.ASA` package:

- `java.sql.Blob`
- `java.sql.Clob`
- `java.sql.Ref`
- `java.sql.Struct`
- `java.sql.Array`
- `java.sql.Map`

The following JDBC 2.0 core functions are not available in the `sybase.sql.ASA` package:

<b>Class in <code>sybase.sql.ASA</code></b>	<b>Missing functions</b>
<code>SAConnection</code>	<code>java.util.Map getTypeMap()</code> <code>void setTypeMap( java.util.Map map )</code>
<code>SAPreparedStatement</code>	<code>void setRef( int pidx, java.sql.Ref r )</code> <code>void setBlob( int pidx, java.sql.Blob b )</code> <code>void setClob( int pidx, java.sql.Clob c )</code> <code>void setArray( int pidx, java.sql.Array a )</code>
<code>SACallableStatement</code>	<code>Object getObject( pidx, java.util.Map map )</code> <code>java.sql.Ref getRef( int pidx )</code> <code>java.sql.Blob getBlob( int pidx )</code> <code>java.sql.Clob getClob( int pidx )</code> <code>java.sql.Array getArray( int pidx )</code>
<code>SAResultSet</code>	<code>Object getObject( int cidx, java.util.Map map )</code> <code>java.sql.Ref getRef( int cidx )</code> <code>java.sql.Blob getBlob( int cidx )</code> <code>java.sql.Clob getClob( int cidx )</code> <code>java.sql.Array getArray( int cidx )</code> <code>Object getObject( String cName, java.util.Map map )</code> <code>java.sql.Ref getRef( String cName )</code> <code>java.sql.Blob getBlob( String cName )</code> <code>java.sql.Clob getClob( String cName )</code> <code>java.sql.Array getArray( String cName )</code>

## Differences between client- and server-side JDBC connections

A difference between JDBC on the client and in the database server lies in establishing a connection with the database environment.

- **Client side** In client-side JDBC, establishing a connection requires the Sybase jConnect JDBC driver. Passing arguments to the `DriverManager.getConnection` establishes the connection. The database environment is an external application from the perspective of the client application.

---

### jConnect required

Depending on the installation package you received, Sybase IQ may or may not include Sybase jConnect. You must have jConnect to use JDBC from external applications. You can use internal JDBC without jConnect.

---

- **Server-side** When using JDBC within the database server, a connection already exists. A value of `jdbc:default:connection` passes to `DriverManager.getConnection`, which provides the JDBC application with the ability to work within the current user connection. This is a quick, efficient and safe operation because the client application has already passed the database security to establish the connection. The user ID and password, having been provided once, do not need to be provided again. The `asajdbc` driver can only connect to the database of the current connection.

You can write JDBC classes in such a way that they can run both at the client and at the server by employing a single conditional statement for constructing the URL. An external connection requires the machine name and port number, while the internal connection requires `jdbc:default:connection`.

## Establishing JDBC connections

This section presents classes that establish a JDBC database connection from a Java application.

## Connecting from a JDBC client application using jConnect

If you wish to access database system tables (database metadata) from a JDBC application, you must add a set of jConnect system objects to your database. Asajdbc shares the same stored procedures for database metadata support with jConnect. These procedures are installed to all databases by default. A dbinit switch “-I” prevents this installation.

For information about adding the jConnect system objects to a database, see “Using the Sybase jConnect JDBC driver” on page 808.

The following complete Java application is a command-line application that connects to a running database, prints a set of information to your command line, and terminates.

Establishing a connection is the first step any JDBC application must take when working with database data.

This example illustrates an external connection, which is a regular client/server connection. For information on how to create an internal connection, from Java classes running inside the database server, see “Establishing a connection from a server-side JDBC class” on page 797.

## External connection example code

The following is the source code for the methods used to make a connection. The source code can be found in the main method and the ASAConnect method of the file *JDBCExamples.java* in the *ASIQ-12\_7/Samples/ASA/JavaSQL/manual-examples* directory on Windows or *ASIQ-12\_7/samples/asa/java* on UNIX under your Sybase IQ installation directory:

```
// Import the necessary classes
import java.sql.*; // JDBC
import com.sybase.jdbc.*; // Sybase jConnect
import java.util.Properties; // Properties
import sybase.sql.*; // Sybase utilities
import asademo.*; // Example classes

private static Connection conn;
public static void main(String args[]) {

    conn = null;
    String machineName;
    if ( args.length != 1 ) {
        machineName = "localhost";
    }
}
```

```
    } else {
        machineName = new String( args[0] );
    }

    ASAConnect( "dba", "sql", machineName );
    if( conn!=null ) {
        System.out.println( "Connection successful" );
    }else{
        System.out.println( "Connection failed" );
    }

    try{
        serializeVariable();
        serializeColumn();
        serializeColumnCastClass();
    }
    catch( Exception e ) {
        System.out.println( "Error: " + e.getMessage() );
        e.printStackTrace();
    }
}
}

private static void ASAConnect( String UserID,
                               String Password,
                               String Machinename ) {
// uses global Connection variable

String _coninfo = new String( Machinename );

Properties _props = new Properties();
_props.put("user", UserID );
_props.put("password", Password );

// Load the Sybase Driver
try {

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

    StringBuffer temp = new StringBuffer();
    // Use the Sybase jConnect driver...
    temp.append("jdbc:sybase:Tds:");
    // to connect to the supplied machine name...
    temp.append(_coninfo);
    // on the default port number for ASA...
    temp.append(":2638");
}
```

```
        // and connect.
        System.out.println(temp.toString());
        conn = DriverManager.getConnection(
            temp.toString() , _props );
    }
    catch ( Exception e ) {
        System.out.println("Error: " + e.getMessage());
        e.printStackTrace();
    }
}
```

## How the external connection example works

The external connection example is a Java command-line application.

### Importing packages

The application requires several libraries, which are imported in the first lines of *JDBCExamples.java*:

- The `java.sql` package contains the Sun Microsystems JDBC classes, which are required for all JDBC applications. You'll find it in the *classes.zip* file in your Java subdirectory.
- Imported from `com.sybase.jdbc`, the Sybase jConnect JDBC driver is required for all applications that connect using jConnect. You'll find it in the *jdbcdrv.zip* file in your Java subdirectory.
- The application uses a property list. The `java.util.Properties` class is required to handle property lists. You'll find it in the *classes.zip* file in your Java subdirectory.
- The `sybase.sql` package contains utilities used for serialization. You'll find it in the *asajdbc.zip* file in your Java subdirectory.
- The `asademo` package contains example classes used in some examples. You'll find it in the *asademo.jar* file in your java subdirectory.

### The main method

Each Java application requires a class with a method named `main`, which is the method invoked when the program starts. In this simple example, `JDBCExamples.main` is the only method in the application.

The `JDBCExamples.main` method carries out the following tasks:

- 1 Processes the command-line argument, using the machine name if supplied. By default, the machine name is *localhost*, which is appropriate for the personal database server.
- 2 Calls the `ASAConnect` method to establish a connection.
- 3 Executes several methods that scroll data to your command line.

The ASAConnect method

The JDBCExamples.ASAConnect method carries out the following tasks:

- 1 Connects to the default running database using Sybase jConnect.
  - Class.forName loads jConnect. Using the newInstance method works around issues in some browsers.
  - The StringBuffer statements build up a connection string from the literal strings and the supplied machine name provided on the command line.
  - DriverManager.getConnection establishes a connection using the connection string.
- 2 Returns control to the calling method.

## Running the external connection example

This section describes how to run the external connection example

❖ **Creating and executing the external connection example application**

- 1 From a system command prompt, change to the Sybase IQ installation directory.
- 2 Change to the *ASIQ-12\_7/java* subdirectory
- 3 Ensure that your CLASSPATH environment variable includes the current directory (.) and the imported zip files. For example, from a command prompt (the following should be entered all on one line):

```
set  
classpath=..\java\jdbcdrv.zip;..;..\java\asajdbc.zip  
;asademo.jar
```

The default zip file name for Java is *classes.zip*. For classes in any file named *classes.zip*, you only need the directory name in the CLASSPATH variable, not the zip-file name itself. For classes in files with other names, you must supply the zip file name.

You need the current directory in the CLASSPATH to run the example.

- 4 Ensure the database is loaded onto a database server running TCP/IP. You can start such a server on your local machine using the following command (from the *ASIQ-12\_7/samples/asa* subdirectory):

On UNIX: `start_asiq .../asiqdemo`

On Windows: `start_asiq ...\\asiqdemo`

- 5 Enter the following at the command prompt to run the example:

```
java JDBCExamples
```

If you wish to try this against a server running on another machine, you must enter the correct name of that machine. The default is `localhost`, which is an alias for the current machine name.

- 6 Confirm that a list of people and products appears at your command prompt.

If the attempt to connect fails, an error message appears instead. Confirm that you have executed all the steps as required. Check that your CLASSPATH is correct. An incorrect CLASSPATH results in a failure to locate a class.

For more information about using jConnect, see “Using the Sybase jConnect JDBC driver” on page 808, and see the online documentation for jConnect.

## Establishing a connection from a server-side JDBC class

SQL statements in JDBC are built using the `createStatement` method of a `Connection` object. Even classes running inside the server need to establish a connection to create a `Connection` object.

Establishing a connection from a server-side JDBC class is more straightforward than establishing an external connection. Because a user already connected executes the server-side class, the class simply uses the current connection.

### Server-side connection example code

The following is the source code for the example. You can find the source code in the `InternalConnect` method of `JDBCExamples.java` in the `ASIQ-12_7/Samples/ASA/JavaSQL/manual-examples` directory under your Sybase IQ installation directory:

```
public static void InternalConnect() {  
    try {  
        conn =  
        DriverManager.getConnection("jdbc:default:connection")  
;  
        System.out.println("Hello World");  
    }  
    catch ( Exception e ) {
```

```
        System.out.println("Error: " + e.getMessage());
        e.printStackTrace();
    }
}
}
```

## How the server-side connection example works

In this simple example, `InternalConnect()` is the only method used in the application.

The application requires only one of the libraries (JDBC) imported in the first line of the `JDBCExamples.java` class. The others are for external connections. The package named `java.sql` contains the JDBC classes.

The `InternalConnect()` method carries out the following tasks:

- 1 Connects to the default running database using the current connection:
  - `DriverManager.getConnection` establishes a connection using a connection string of `jdbc:default:connection`.
- 2 Prints `Hello World` to the current standard output, which is the server window. `System.out.println` carries out the printing.
- 3 If there is an error in the attempt to connect, an error message appears in the server window, together with the place where the error occurred.  
The try and catch instructions provide the framework for the error handling.
- 4 The class terminates.

## Running the server-side connection example

This section describes how to run the server-side connection example.

### ❖ Creating and executing the internal connection example application

- 1 If you have not already done so, compile the `JDBCExamples.java` file. If you are using the JDK, you can do the following in the `ASIQ-12_7/Samples/ASA/JavaSQL/manual-examples` directory under your Sybase IQ installation directory from a command prompt:

```
javac JDBCExamples.java
```

- 2 Start a database server using the sample database. You can start such a server on your local machine using the following command (from the `/ASIQ-12_7/java` subdirectory):

On UNIX: `start_asiq .../asiqdemo`

On Windows: `start_asiq ...\\asiqdemo`

The TCP/IP network protocol is not necessary in this case, since you are not using jConnect. However, you must have at least 8 Mb of cache available to use Java classes in the database.

- 3 Install the class into the sample database. Once connected to the sample database, you can do this from Interactive SQL using the following command:

```
INSTALL JAVA NEW
FROM FILE
'path\\\\ASIQ-12_7\\Samples\\ASA\\JavaSQL\\manual-
examples\\JDBCExamples.class'
```

where `path` is the path to your installation directory.

You can also install the class using Sybase Central. While connected to the sample database, open the Java Objects folder and double-click Add Class. Then follow the instructions in the wizard.

- 4 You can now call the `InternalConnect` method of this class just as you would a stored procedure:

```
CALL JDBCExamples>>InternalConnect()
```

The first time a Java class is called in a session, the internal Java virtual machine must be loaded. This can take a few seconds.

- 5 Confirm that the message `Hello World` prints on the server screen.

## Notes on JDBC connections

- **Autocommit behavior** The JDBC specification requires that, by default, a COMMIT is performed after each data modification statement. Currently, the server-side JDBC behavior is to commit. You can control this behavior using a statement such as the following:

```
conn.setAutoCommit( false ) ;
```

where `conn` is the current connection object.

- **Connection defaults** From server-side JDBC, only the first call to `getConnection( "jdbc:default:connection" )` creates a new connection with the default values. Subsequent calls return a wrapper of the current connection with all connection properties unchanged. If you set AutoCommit to OFF in your initial connection, any subsequent `getConnection` calls within the same Java code return a connection with AutoCommit set to OFF.

You may wish to ensure that closing a connection resets connection properties to their default values, so subsequent connections are obtained with standard JDBC values. The following type of code achieves this:

```
Connection conn = DriverManager.getConnection( "" );
boolean oldAutoCommit = conn.getAutoCommit();
try {
    // do code here
}
finally {
    conn.setAutoCommit( oldAutoCommit );
}
```

This discussion applies not only to AutoCommit, but also to other connection properties such as TransactionIsolation and is ReadOnly.

## Using JDBC to access data

Java applications that hold some or all classes in the database have significant advantages over traditional SQL stored procedures. At an introductory level, however, it may be helpful to use the parallels with SQL stored procedures to demonstrate the capabilities of JDBC. In the following examples, we write Java classes that insert a row into the Department table.

As with other interfaces, SQL statements in JDBC can be either **static** or **dynamic**. Static SQL statements are constructed in the Java application, and sent to the database. The database server parses the statement, and selects an execution plan, and executes the statement. Together, parsing and selecting an execution plan are referred to as **preparing** the statement.

If a similar statement has to be executed many times (many inserts into one table, for example), there can be significant overhead in static SQL because the preparation step has to be executed each time.

In contrast, a dynamic SQL statement contains placeholders. The statement, prepared once using these placeholders, can be executed many times without the additional expense of preparing.

In this section we use static SQL. Dynamic SQL is discussed in a later section.

## Preparing for the examples

This section describes how to prepare for the examples in the remainder of this appendix.

### Sample code

The code fragments in this section are taken from the complete class *ASIQ-12\_7\Samples\ASA\JavaSQL\manual-examples\JDBCEExamples.java*, under your installation directory.

#### ❖ Installing the **JDBCEExamples** class

- If you have not already done so, install the *JDBCEExamples.class* file into the sample database. Once connected to the sample database from Interactive SQL, enter the following command in the SQL Statements pane:

```
INSTALL JAVA NEW
FROM FILE '\\path\ASIQ-
12_7\Samples\ASA\JavaSQL\manual-
examples\JDBCEExamples.class'
```

where *path* is the path to your installation directory.

You can also install the class using Sybase Central. While connected to the sample database, open the Java Objects folder and double-click Add Java Class or JAR. Then follow the instructions in the wizard.

## Inserts, updates, and deletes using JDBC

The Statement object executes static SQL statements. You execute SQL statements such as INSERT, UPDATE, and DELETE, which do not return result sets, using the *executeUpdate* method of the Statement object. Statements such as CREATE TABLE and other data definition statements can also be executed using *executeUpdate*.

The following code fragment illustrates how JDBC carries out INSERT statements. It uses an internal connection held in the Connection object named conn. The code for inserting values from an external application using JDBC would need to use a different connection, but otherwise would be unchanged.

```
public static void InsertFixed() {  
    // returns current connection  
    conn = DriverManager.getConnection("jdbc:default:connection");  
    // Disable autocommit  
    conn.setAutoCommit( false );  
  
    Statement stmt = conn.createStatement();  
  
    Integer IRows = new Integer( stmt.executeUpdate  
        ("INSERT INTO Department (dept_id, dept_name )"  
        + "VALUES (201, 'Eastern Sales')"  
        ) );  
    // Print the number of rows updated  
    System.out.println(IRows.toString() + "row inserted" );  
}
```

---

#### Source code available

This code fragment is part of the `InsertFixed` method of the `JDBCExamples` class, included in the `ASIQ-12_7\Samples\ASA\JavaSQL\manual-examples` subdirectory of your installation directory.

---

#### Notes

- The `setAutoCommit` method turns off the AutoCommit behavior, so changes are only committed if you execute an explicit `COMMIT` instruction.
- The `executeUpdate` method returns an integer, which reflects the number of rows affected by the operation. In this case, a successful `INSERT` would return a value of one (1).
- The integer return type converts to an `Integer` object. The `Integer` class is a wrapper around the basic `int` data type, providing some useful methods such as `toString()`.
- The `Integer` `IRows` converts to a string to be printed. The output goes to the server window.

#### ❖ Running the JDBC Insert example

- 1 Using Interactive SQL, connect to the sample database as user ID `dba`.
- 2 Ensure the `JDBCExamples` class has been installed. It is installed together with the other Java examples classes.

For instructions on installing the Java examples classes, see “Preparing for the examples” on page 801.

- 3 Call the method as follows:

```
CALL JDBCExamples>>InsertFixed()
```

- 4 Confirm that a row has been added to the department table.

```
SELECT *
FROM department
```

The row with ID 201 is not committed. You can execute a ROLLBACK statement to remove the row.

In this example, you have seen how to create a very simple JDBC class. Subsequent examples expand on this.

## Passing arguments to Java methods

We can expand the InsertFixed method to illustrate how arguments are passed to Java methods.

The following method uses arguments passed in the call to the method as the values to insert:

```
public static void InsertArguments(
    String id, String name) {
try {
    conn = DriverManager.getConnection(
        "jdbc:default:connection" );

    String sqlStr = "INSERT INTO Department "
        + " ( dept_id, dept_name )"
        + " VALUES (" + id + ", '" + name + "' )";

    // Execute the statement
    Statement stmt = conn.createStatement();
    Integer IRows = new Integer( stmt.executeUpdate(
        sqlStr.toString() ) );

    // Print the number of rows updated
    System.out.println( IRows.toString() + " row
inserted" );
}
catch ( Exception e ) {
    System.out.println("Error: " + e.getMessage());
}
```

```
        e.printStackTrace();
    }
}
```

Notes

- The two arguments are the department id (an integer) and the department name (a string). Here, both arguments pass to the method as strings, because they are part of the SQL statement string.
- The INSERT is a static statement and takes no parameters other than the SQL itself.
- If you supply the wrong number or type of arguments, you receive the `Procedure Not Found` error.

❖ **Using a Java method with arguments**

- 1 If you have not already installed the `JDBCEexamples.class` file into the sample database, do so.
- 2 Connect to the sample database from Interactive SQL, and enter the following command:

```
call JDBCEexamples>>InsertArguments( '203',
'Northern Sales' )
```

- 3 Verify that an additional row has been added to the Department table:

```
SELECT *
FROM Department
```

- 4 Roll back the changes to leave the database unchanged:

```
ROLLBACK
```

## Queries using JDBC

The Statement object executes static queries, as well as statements that do not return result sets. For queries, you use the `executeQuery` method of the Statement object. This returns the result set in a `ResultSet` object.

The following code fragment illustrates how queries can be handled within JDBC. The code fragment places the total inventory value for a product into a variable named `inventory`. The product name is held in the `String` variable `prodname`. This example is available as the `Query` method of the `JDBCEexamples` class.

The example assumes an internal or external connection has been obtained and is held in the `Connection` object named `conn`. It also assumes a variable

```

public static void Query () {
    int max_price = 0;
    try{
        conn = DriverManager.getConnection(
            "jdbc:default:connection" );

        // Build the query
        String sqlStr = "SELECT id, unit_price "
        + "FROM product" ;

        // Execute the statement
        Statement stmt = conn.createStatement();
        ResultSet result = stmt.executeQuery( sqlStr );

        while( result.next() ) {
            int price = result.getInt(2);
            System.out.println( "Price is " + price );
            if( price > max_price ) {
                max_price = price ;
            }
        }
    } catch( Exception e ) {
        System.out.println("Error: " + e.getMessage());
        e.printStackTrace();
    }
    return max_price;
}

```

**Running the example**

Once you have installed the JDBCExamples class into the sample database, you can execute this method using the following statement in Interactive SQL:

```
select JDBCExamples>>Query()
```

**Notes**

- The query selects the quantity and unit price for all products named prodname. These results are returned into the ResultSet object named result.
- There is a loop over each of the rows of the result set. The loop uses the next method.
- For each row, the value of each column is retrieved into an integer variable using the getInt method. ResultSet also has methods for other data types, such as getString, getDate, and getBinaryString.

The argument for the getInt method is an index number for the column, starting from 1.

Data type conversion from SQL to Java is carried out according to the information in “Java / SQL data type conversion” in the “SQL Data Types” chapter of the *Sybase IQ Reference Manual*.

- Sybase IQ supports bidirectional scrolling cursors. However, JDBC provides only the next method, which corresponds to scrolling forward through the result set.
- The method returns the value of max\_price to the calling environment, and Interactive SQL displays it in the Results pane.

## Using prepared statements for more efficient access

If you use the Statement interface, you parse each statement you send to the database, generate an access plan, and execute the statement. The steps prior to actual execution are called **preparing** the statement.

You can achieve performance benefits if you use the PreparedStatement interface. This allows you to prepare a statement using placeholders, and then assign values to the placeholders when executing the statement.

Using prepared statements is particularly useful when carrying out many similar actions, such as inserting many rows.

For more information on prepared statements, see the “PREPARE statement [ESQL]” in Chapter 6, “SQL Statements,” in the *Sybase IQ Reference Manual*.

### Example

The following example illustrates how to use the PreparedStatement interface, although inserting a single row is not a good use of prepared statements.

The following method of the JDBCExamples class carries out a prepared statement:

```
public static void JInsertPrepared(int id, String name)
try {
    conn = DriverManager.getConnection(
        "jdbc:default:connection");

    // Build the INSERT statement
    // ? is a placeholder character
    String sqlStr = "INSERT INTO Department "
    + "( dept_id, dept_name ) "
    + "VALUES ( ? , ? )";

    // Prepare the statement
    PreparedStatement stmt = conn.prepareStatement(
```

```
sqlStr ) ;

stmt.setInt(1, id);
stmt.setString(2, name );
Integer IRows = new Integer(
                      stmt.executeUpdate() ) ;

// Print the number of rows updated
System.out.println(IRows.toString() + " row
inserted" );
}
catch ( Exception e ) {
    System.out.println("Error: " + e.getMessage());
    e.printStackTrace();
}
}
```

#### Running the example

Once you have installed the JDBCExamples class into the sample database, you can execute this example by entering the following statement:

```
call JDBCExamples>>InsertPrepared(
                           202, 'Eastern Sales' )
```

The string argument is enclosed in single quotes, which is appropriate for SQL. If you invoked this method from a Java application, use double quotes to delimit the string.

## Inserting and retrieving objects

As an interface to relational databases, JDBC is designed to retrieve and manipulate traditional SQL data types. Sybase IQ also provides abstract data types in the form of Java classes. The way you access these Java classes using JDBC depends on whether you want to insert or retrieve the objects.

For more information on getting and setting entire objects, see “Creating distributed applications” on page 813.

## Retrieving objects

You can retrieve objects and their fields and methods by:

- **Accessing methods and fields** Java methods and fields can be included in the select-list of a query. A method or field then appears as a column in the result set, and can be accessed using one of the standard ResultSet methods, such as getInt, or getString.

- **Retrieving an object** If you include a column with a Java class data type in a query select list, you can use the ResultSet getObject method to retrieve the object into a Java class. You can then access the methods and fields of that object within the Java class. Java objects can only be stored in the Catalog Store.

## Inserting objects

From a server-side Java class, you can use the JDBC setObject method to insert an object into a column with Java class data type.

You can insert objects using a prepared statement. For example, the following code fragment inserts an object of type MyJavaClass into a column of table T:

```
java.sql.PreparedStatement ps =
    conn.prepareStatement("insert T values( ? )" );
ps.setObject( 1, new MyJavaClass() );
ps.executeUpdate();
```

An alternative is to set up a SQL variable that holds the object and then to insert the SQL variable into the table.

## Miscellaneous JDBC notes

- **Access permissions** Like all Java classes in the database, classes containing JDBC statements can be accessed by any user. There is no equivalent of the GRANT EXECUTE statement that grants permission to execute procedures, and there is no need to qualify the name of a class with the name of its owner.
- **Execution permissions** Java classes are executed with the permissions of the connection executing them. This behavior is different to that of stored procedures, which execute with the permissions of the owner.

# Using the Sybase jConnect JDBC driver

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

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

For a full description of jConnect and its use with Sybase IQ, see the jConnect documentation available in the online books or from the jConnect web site at <http://www.sybase.com/products/eaimiddleware/jconnectforjdbc/>

## Versions of jConnect supplied with Sybase IQ

Sybase IQ provides the following versions of the Sybase jConnect JDBC driver:

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

## The jConnect driver files

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

Classpath setting for jConnect

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

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

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

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

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

Importing the  
jConnect classes

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

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

## Installing jConnect system objects into a database

If you wish to use jConnect to access system table information (database metadata), you must add the jConnect system objects to your database.

By default, the jConnect system objects are added to a database for any database created using version 12.7, and to any database upgraded to version 12.7.

You can choose to add the jConnect objects to the database either when creating or upgrading, or at a later time.

You can install the jConnect system objects from Interactive SQL.

❖ **Adding the jConnect system objects to a version 12.7 database from Interactive SQL**

- Connect to the database from Interactive SQL as a user with DBA authority, and enter the following command in the SQL Statements pane:

```
read path\scripts\jcatalog.sql
```

where *path* is your Sybase IQ installation directory.

---

**Tip**

You can also use a command prompt to add the jConnect system objects to a version 12.7 database. At the command prompt, type:

```
dbisql -c "uid=user;pwd=pwd" path\scripts\jcatalog.sql
```

where *user* and *pwd* identify a user with DBA authority, and *path* is your Sybase IQ installation directory.

---

## Loading the driver

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

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

Using the newInstance method works around issues in some browsers.

## Supplying a URL for the server

To connect to a database via jConnect, you need to supply a Universal Resource Locator (URL) for the database. An example given in the section is as follows:

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

The URL is put together in the following way:

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

The individual components are include:

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

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

## Specifying a database on a server

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

Using the  
ServiceName  
parameter

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

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

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

Using the  
RemotePWD  
parameter

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

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

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

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

For information on the utility database, see “The utility database” on page 26 and “Utility database server security” on page 564.

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

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

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

where myconnection becomes the IQ connection name.

# Creating distributed applications

In a **distributed application**, parts of the application logic run on one machine, and parts run on another machine. With Sybase IQ, you can create distributed Java applications, where part of the logic runs in the database server, and part on the client machine.

Sybase IQ is capable of exchanging Java objects with an external, Java client.

Having the client application retrieve a Java object from a database is the key task in a distributed application. This section describes how to accomplish that task.

## Related tasks

In other parts of this appendix, we describe several tasks related to retrieving objects, but these tasks should not be confused with retrieving the object itself. For example:

- “Queries using JDBC” on page 804 describes how to retrieve an object into a SQL variable. This does not solve the problem of getting the object into your Java application.
- “Queries using JDBC” on page 804 also describes how to retrieve the public fields and the return value of Java methods. Again, this is distinct from retrieving an object into a Java application.
- “Inserting and retrieving objects” on page 807 describes how to retrieve objects into server-side Java classes. Again, this is not the same as retrieving them into a client application.

## Requirements for distributed applications

There are two tasks in building a distributed application.

### ❖ Building a distributed application

- 1 Any class running in the server must implement the Serializable interface. This is very simple.
- 2 The client-side application must import the class, so the object can be reconstructed on the client side.

These tasks are described in the following sections.

## Implementing the Serializable interface

Objects pass from the server to a client application in **serialized** form. For an object to be sent to a client application, it must implement the Serializable interface. Fortunately, this is a very simple task.

### ❖ Implementing the Serializable interface

- Add the words implements java.io.Serializable to your class definition.

---

***Reviewers: ASIQ-12\_7\Samples\ASA\JavaSQL\manual-examples\, but I don't see the Product class at all.***

---

For example, the Product class in the in  
\$ASDIR/samples/asa/java/asademo (UNIX) or  
%ASDIR%\samples\asa\java\asademo (Windows) subdirectory  
implements the Serializable interface by virtue of the following  
declaration:

```
public class Product implements java.io.Serializable
```

Implementing the Serializable interface amounts to simply declaring that  
your class can be serialized.

The Serializable interface contains no methods and no variables. Serializing an  
object converts it into a byte stream which allows it to be saved to disk or sent  
to another Java application where it can be reconstituted, or **deserialized**.

A serialized Java object in a database server, sent to a client application and  
deserialized, is identical in every way to its original state. Some variables in an  
object, however, either don't need to be or, for security reasons, should not be  
serialized. Those variables are declared using the keyword transient, as in the  
following variable declaration.

```
transient String password;
```

When an object with this variable is deserialized, the variable always contains  
its default value, null.

Custom serialization can be accomplished by adding writeObject() and  
readObject() methods to your class.

For more information about serialization, see Sun Microsystems' Java  
Development Kit (JDK).

## Importing the class on the client side

On the client side, any class that retrieves an object has to have access to the proper class definition to use the object. To use the Product class, which is part of the `asademo` package, you must include the following line in your application:

```
import asademo.*
```

The `asademo.jar` file must be included in your CLASSPATH for this package to be located.

## A sample distributed application

The `JDBCExamples.java` class contains three methods that illustrate distributed Java computing. These are all called from the main method. This method is called in the connection example described in “Connecting from a JDBC client application using jConnect” on page 793, and is an example of a distributed application.

Here is the `getObjectColumn` method from the `JDBCExamples` class.

```
private static void getObjectColumn() throws Exception
{
    // Return a result set from a column containing
    // Java objects
    asademo.ContactInfo ci;
    String name;
    String sComment ;

    if ( conn != null ) {
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery(
            "SELECT JContactInfo FROM jdba.contact"
        );
        while ( rs.next() ) {
            ci = ( asademo.ContactInfo )rs.getObject(1);
            System.out.println( "\n\tStreet: " + ci.street +
                "City: " + ci.city +
                "\n\tState: " + ci.state +
                "Phone: " + ci.phone +
                "\n" );
        }
    }
}
```

The `getObject` method is used in the same way as in the internal Java case.

## Other features of distributed applications

There are two other methods in *JDBCExamples.java* that use distributed computing:

- **serializeVariable** This method creates a native Java object referenced by a SQL variable on the database server and passes it back to the client application.
- **serializeColumnCastClass** This method is like the `serializeColumn` method, but demonstrates how to reconstruct subclasses. The column that is queried (`JProd` from the `product` table) is of data type `asademo.Product`. Some of the rows are `asademo.Hat`, which is a subclass of the `Product` class. The proper class is reconstructed on the client side.

# Debugging Logic in the Database

## About this appendix

This appendix tells you how to use the Sybase debugger to assist in developing SQL stored procedures and event handlers, as well as Java stored procedures.

## Contents

Topic	Page
Introduction to debugging in the database	817
Tutorial 1: Getting started with the debugger	819
Tutorial 2: Debugging a stored procedure	820
Tutorial 3: Debugging a Java class	823
Common debugger tasks	827
Writing debugger scripts	829

## Introduction to debugging in the database

You can use the debugger during the development of the following objects:

- SQL stored procedures, event handlers, and user-defined functions.
- Java stored procedures in the database.

This appendix tells you how to set up and use the debugger.

## Debugger features

You can carry out many tasks with the Sybase debugger, including the following:

- **Debug procedures** You can debug SQL stored procedures.

- **Debug event handlers** Event handlers are an extension of SQL stored procedures. The material in this chapter about debugging stored procedures applies equally to debugging event handlers.
- **Browse classes and stored procedures** You can browse through the source code of installed classes and SQL procedures.
- **Debug Java classes** You can debug Java classes that are stored in the database.
- **Trace execution** Step line by line through the code of a Java class or stored procedure running in the database. You can also look up and down the stack of functions that have been called.
- **Set breakpoints** Run the code until you hit a breakpoint, and stop at that point in the code.
- **Set break conditions** Breakpoints include lines of code, but you can also specify conditions when the code is to break. For example, you can stop at a line the tenth time it is executed, or only if a variable has a particular value. You can also stop whenever a particular exception is thrown in a Java application.
- **Inspect and modify local variables** When execution is stopped at a breakpoint, you can inspect the values of local variables and alter their value.
- **Inspect and break on expressions** When execution is stopped at a breakpoint, you can inspect the value of a wide variety of expressions.
- **Execute queries** When execution is stopped at a breakpoint in a SQL procedure, you can execute queries. This permits you to look at intermediate results held in temporary tables, as well as checking values in base tables.

## Requirements for using the debugger

You need the following in order to use the debugger:

- **Permissions** In order to use the debugger, you must either have DBA authority or be granted permissions in the SA\_DEBUG group. This group is added to all databases when they are created.
- **Source code for Java classes** The source code for your application must be available to the debugger. For Java classes, the source code is held on a directory on your hard disk. For stored procedures, the source code is held in the database.

- **Compilation options** To debug Java classes, they must be compiled so that they contain debugging information. For example, if you are using the Sun Microsystems JDK compiler *javac.exe*, they must be compiled using the *-g* command-line option.

## Tutorial 1: Getting started with the debugger

This tutorial describes how to start the debugger, how to connect to a database, and how to debug a Java class.

### Lesson 1: Connect to a database and start the debugger

This tutorial shows you how to start the debugger, connect to a database, and attach to a connection for debugging. It uses the Sybase IQ sample database.

#### Start the debugger

##### ❖ Starting the debugger

###### 1 Start Sybase Central:

Choose Start > Programs > Adaptive Server IQ 12.6 > Sybase Central Java Edition.

###### 2 Connect to the database.

In this tutorial, you connect to the Sybase IQ sample database.

- In the left pane, right click Sybase IQ and choose Connect from the popup menu.

The connection dialog appears.

###### 3 Choose Debug mode.

Sybase Central can be used in Design mode or Debug mode. When running in Debug mode, debugger breakpoints are active. Also, Sybase Central shows debugging menus and a Debugger Details pane.

To choose Debug mode, click Task > Debug. The Debugger Details pane appears at the bottom of Sybase Central and the Sybase IQ toolbar displays a set of debugger tools.

## Tutorial 2: Debugging a stored procedure

This tutorial describes a sample session for debugging a stored procedure. It is a continuation of “Tutorial 1: Getting started with the debugger” on page 819.

In this tutorial, you call the stored procedure `sp_customer_products`, which is part of the sample database.

The `sp_customer_products` procedure executes the following query against the sample database:

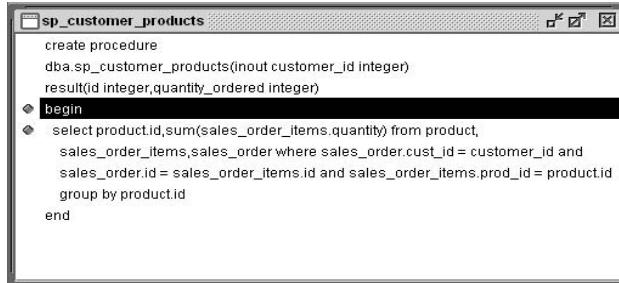
```
CREATE PROCEDURE dba.sp_customer_products(
    INOUT customer_id INTEGER )
RESULT(id integer,quantity_ordered integer)
BEGIN
    SELECT product.id,sum(sales_order_items.quantity)
    FROM product,sales_order_items,sales_order
    WHERE sales_order.cust_id = customer_id
    AND sales_order.id = sales_order_items.id
    AND sales_order_items.prod_id = product.id
    GROUP BY product.id
END
```

It takes a customer ID as input, and returns as a result set a list of product IDs and the number ordered by that customer.

## Display stored procedure source code in the debugger

The stored procedure source code is stored in the database. You can display the stored procedure source code in the Source window.

- ❖ **Displaying stored procedure source code in the debugger**
  - From the debugger interface, double-click the sp\_customer\_products stored procedure. The source code for the procedure appears in the Source window.



The screenshot shows the Oracle SQL Developer interface with a Source Code window titled "sp\_customer\_products". The code listed is:

```

create procedure
dba.sp_customer_products(inout customer_id integer)
result(id integer,quantity_ordered integer)
begin
  select product.id,sum(sales_order_items.quantity) from product,
  sales_order_items,sales_order where sales_order.cust_id = customer_id and
  sales_order.id = sales_order_items.id and sales_order_items.prod_id = product.id
  group by product.id
end

```

## Set a breakpoint

You can set a breakpoint in the body of the sp\_customer\_products procedure. When the procedure is executed, execution stops at the breakpoint.

- ❖ **Setting a breakpoint in a stored procedure**

- 1 In the Source Code window, locate the line with the query:

```
select product.id,...
```

- 2 Click the green indicator to the left of the line, until it is red.

Repeatedly clicking the indicator toggles its status.

## Run the procedure

You can call the stored procedure from Interactive SQL (DBISQL), and see its execution interrupted at the breakpoint.

- ❖ **Calling the procedure from Interactive SQL**

- 1 Start Interactive SQL. Connect to the sample database with a user ID of DBA and a password of SQL.

The connection appears in the debugger Connections window list.

- 2 Enter the following command in Interactive SQL to call the procedure using the customer with ID 122:

```
CALL sp_customer_products( 122 )
```

The query does not complete. Instead, execution is stopped in the debugger at the breakpoint. In Interactive SQL, the Interrupt the SQL Statement button is active. In the debugger Source window, the red arrow indicates the current line.

- 3 Step to the next line by choosing Run > Step Over. You can also press F7.

For longer procedures, you can use other methods of stepping through code. For more examples, see “Tutorial 3: Debugging a Java class” on page 823.

For the next lesson, leave the debugger stopped at the SELECT line.

## Inspect and modify variables

You can inspect the values of variables in the debugger.

### Inspecting local variables

You can inspect the values of local variables in a procedure as you step through the code, to better understand what is happening.

#### ❖ Inspecting and modifying the value of a variable

- 1 If the Local Variables window is not displayed, choose Window > Local Variables to display it.

The Local Variables window shows that there are two local variables; the stored procedure itself (which does not have a return value, and so is listed as NULL) and the customer\_id passed in to the procedure.

- 2 In the Local Variables window, double-click the Value column entry for customer\_id, and type in 125 to change the customer ID value used in the query.
- 3 In the Source window, press F5 to complete the execution of the query and finish the tutorial.

The Interactive SQL Results window displays the list of product IDs and quantities for customer 125:

<b>Id</b>	<b>sum(sales_order....)</b>
301	60
700	48
...	...

## Tutorial 3: Debugging a Java class

This tutorial describes a sample session for debugging a stored procedure. It is a continuation of “Tutorial 1: Getting started with the debugger” on page 819.

In this tutorial, you call `JDBCExamples.Query()` from Interactive SQL (DBISQL), interrupt the execution in the debugger, and trace through the source code for this method.

The `JDBCExamples.Query()` method executes the following query against the sample database:

```
SELECT id, unit_price  
FROM product
```

It then loops through all the rows of the result set, and returns the one with the highest unit price.

Compiling Java classes for debugging

You must compile classes with the `javac -g` option in order to debug them. The sample classes are compiled for debugging.

## Prepare the database

If you intend to run Java examples, such as, you need to install the Java example classes into the sample database.

For more information about the `JDBCExamples` class and its methods, see “Using JDBC to access data” on page 800.

## Display Java source code into the debugger

The debugger looks in a set of locations for source code files (with `.java` extension). You need to add the `samples/asa/java` subdirectory (`Samples\ASA\JavaSQL\manual-examples` on Windows) of your installation directory to the list of locations, so that the code for the class currently being executed in the database is available to the debugger.

### ❖ Displaying Java source code in the debugger

- 1 From the debugger interface, select File > Edit Source Path. The Source path window appears.

- 2 Enter the path to the `/asa/java` subdirectory of your Sybase IQ installation directory. For example, if you installed Sybase IQ in `%ASDIR%`, you would enter the following:

%ASDIR%\asa\java

- 3 Click Apply, and close the window.
  - 4 In the Classes window, double-click JDBCExamples. The source code for the JDBCExamples class appears in the Source window.

```
JDBCExamples
◆ import java.sql.*;           // JDBC
◆ import com.sybase.jdbc.*;    // Sybase jConnect
◆ import java.util.Properties; // Properties
◆ import sybase.sql.*;        // Sybase utilities
◆ import asademo.*;          // Example classes
◆
◆ public class JDBCExamples{
◆     private static Connection conn;
◆
◆     public static void main( String args[]){
◆         // Establish a connection
◆         conn = null;
◆         String machineName = ( args.length == 1 ? args[0] : "localhost" );
◆         ASAConnect("dba", "sql", machineName );
◆         if( conn!=null ) {
◆             System.out.println("Connection successful");
◆         }else{
◆             System.out.println("Connection failed");
◆         }
◆     }
◆
◆     try{
◆         getObjectColumn0();
◆     }
```

## Notes on locating Java source code

The Source Path window holds a list of directories in which the debugger looks for Java source code. Java rules for finding packages apply. The debugger also searches the current CLASSPATH for source code.

For example, on Windows, if you add the paths %ASDIR%\asa\java and c:\Java\src to the source path, and the debugger is trying to find a class called *asademo.Product*, it looks for the source code in c:\Program Files\Sybase\ASIQ-12\_6\asa\java\asademo\Product.Java and c:\Java\src\my\asademo\Product.Java

## Set a breakpoint

You can set a breakpoint at the beginning of the `Query()` method. When the method is invoked, execution stops at the breakpoint.

#### ❖ Setting a breakpoint in a Java class

- 1 In the Source Code window, page down until you see the beginning of the Query() method. This method is near the end of the class, and starts with the following line:

```
public static int Query() {
```

- 2 Click the green indicator to the left of the first line of the method, until it is red. The first line of the method is:

```
int max_price = 0;
```

Repeatedly clicking the indicator toggles its status.

## Run the method

You can invoke the Query() method from Interactive SQL (DBISQL), and see its execution interrupted at the breakpoint.

#### ❖ Invoking the method from Interactive SQL

- 1 Start Interactive SQL. Connect to the sample database as user ID DBA and password SQL.

The connection appears in the debugger Connections window list.

- 2 Enter the following command in Interactive SQL to invoke the method:

```
SELECT JDBCExamples.Query()
```

The query does not complete. Instead, execution is stopped in the debugger at the breakpoint. In Interactive SQL, the Stop button is active. In the debugger Source window, the red arrow indicates the current line.

You can now step through source code and carry out debugging activities in the debugger.

## Step through source code

In this section we illustrate some of the ways you can step through code in the debugger.

Following the previous section, the debugger should have stopped execution of JDBCExamples.Query() at the first statement in the method:

Examples

Here are some example steps you can try:

- 1 Step to the next line** Choose Run > Step Over, or press F7 to step to the next line in the current method. Try this two or three times.

- 2 Run to a selected line** Select the following line using the mouse, and choose Run > Run To Selected, or press F6 to run to that line and break:

```
max_price = price;
```

The red arrow moves to the line.

- 3 Set a breakpoint and execute to it** Select the following line (line 292) and press F9 to set a breakpoint on that line:

```
return max_price;
```

An asterisk appears in the left hand column to mark the breakpoint. Press F5 to execute to that breakpoint.

- 4 Experiment** Try different methods of stepping through the code. End with F5 to complete the execution.

When you have completed the execution, the Interactive SQL Data window displays the value 24.

#### Options

The complete set of options for stepping through source code are displayed on the Run menu. You can find more information in the debugger online Help.

## Inspect and modify variables

You can inspect the values of both local variables (declared in a method) and class static variables in the debugger.

#### Inspecting local variables

You can inspect the values of local variables in a method as you step through the code, to better understand what is happening. You must have compiled the class with the *javac -g* option to do this.

#### ❖ Inspecting and modifying the value of a variable

- 1 Set a breakpoint at the first line of the JDBCExamples.Query method. This line is as follows:

```
int max_price = 0
```

- 2 In Interactive SQL, enter the following statement again to execute the method:

```
SELECT JDBCExamples.Query()
```

The query executes only as far as the breakpoint.

- 3 Press F7 to step to the next line. The `max_price` variable has now been declared and initialized to zero.
  - 4 If the Local Variables window is not displayed, choose Window > Local Variables to display it.
- The Local Variables window shows that there are several local variables. The `max_price` variable has a value of zero. All others are listed as `variable not in scope`, which means they are not yet initialized.
- 5 In the Local Variables window, double-click the Value column entry for `max_price`, and type in 45 to change the value of `max_price` to 45.
- The value 45 is larger than any other price. Instead of returning 24, the query will now return 45 as the maximum price.
- 6 In the Source window, press F7 repeatedly to step through the code. As you do so, the values of the variables appear in the Local Variables window. Step through until the `stmt` and `result` variables have values.
  - 7 Expand the `result` object by clicking the icon next to it, or setting the cursor on the line and pressing ENTER. This displays the values of the fields in the object.
  - 8 When you have experimented with inspecting and modifying variables, press F5 to complete the execution of the query and finish the tutorial.

#### Inspecting static variables

In addition to local variables, you can display class-level variables (static variables) in the debugger Statics window, and inspect their values in the Inspection window. For more information, see the debugger online Help.

## Common debugger tasks

If you want to...	Then choose...
Add a directory to the Java source file search path	File > Add Source Path
Display callee's context	Stack > Down
Display caller's context	Stack > Up
Enable capturing of connections by the debugger	Connection > Enable capture
Exit the debugger	File > Exit
Find a string in the selected window	Search > Find

If you want to...	Then choose...
Ignore the case of a word when searching	Search > Ignore Case
Load the procedure debugger's settings from a file	Settings > Load From File
Login to the database as a new connection	Connection > Login
Log out of the database	Connection > Logout
Restart a program's execution	Run > Restart
Run a debug script	File > Run Script For more information, see “Writing debugger scripts” on page 829
Run a program line by line, going through procedures line by line as well	Run > Step Into
Run a program, line by line, switching between a Java program and stored procedures in other environments, when applicable	Run > Step Through
Run a program, line by line, without going through procedures line by line. If any procedure contains a breakpoint, the breakpoint will be ignored.	Run > Step Over
Run a program, or resume a program's execution from a breakpoint	Run > Go
Run a program until the current procedure/method returns.	Run > Step Out
Save procedures' breakpoints within procedures	Settings > Remember Breakpoints
Save the procedure debugger's settings	Settings > Save
Save the procedure debugger's settings to a file	Settings > Save to File
Save the procedure debugger's window positions, fonts, etc. automatically upon exiting the debugger	Settings > Save on Exit
Save the procedure debugger's window positions, fonts, etc. with settings.	Settings > Remember Windows Attributes

If you want to...	Then choose...
Specify the path containing the Java source file	File > Edit Source Path
View the source code for a procedure or class	File > Open

## Writing debugger scripts

The debugger allows you to write scripts in the Java programming language. A script is a Java class that extends the “sybase.asa.procdebug.DebugScript class” on page 829.

When the debugger runs a script, it loads the class and calls its run method. The first parameter of the run method is a pointer to an instance of the. This interface lets you interact with and control the debugger.

A debugger window is represented by the “sybase.asa.procdebug.IDebugWindow interface” on page 834.

You can compile scripts with a command such as the following:

```
javac -classpath
%asany%/procdebug/ProcDebug.jar;%classpath%
myScript.Java.
```

## sybase.asa.procdebug.DebugScript class

You can write scripts to control debugger behavior. Scripts are classes that extend the DebugScript class.

The DebugScript class is as follows:

```
// All debug scripts must inherit from this class

package sybase.asa.procdebug;

abstract public class DebugScript
{
    abstract public void run( IDebugAPI db, String args []
);
/*
The run method is called by the debugger
```

```
    - args will contain command line arguments
    */

    public void OnEvent( int event ) throws DebugError {}
    /*
     - Override the following methods to process debug
     events
     - NOTE: this method will not be called unless you
     call
         DebugAPI.AddEventHandler( this );
    */
}

}
```

## **sybase.asa.procdebug.IDebugAPI interface**

You can write scripts to control debugger behavior. Scripts are Java classes that use the IDebugAPI interface to control the debugger.

The IDebugAPI interfaces is as follows:

```
package sybase.asa.procdebug;
import java.util.*;
public interface IDebugAPI

{
    // Simulate Menu Items

    IDebugWindow MenuOpenSourceWindow() throws
    DebugError;
    IDebugWindow MenuOpenCallsWindow() throws
    DebugError;
    IDebugWindow MenuOpenClassesWindow() throws
    DebugError;
    IDebugWindow MenuOpenClassListWindow() throws
    DebugError;
    IDebugWindow MenuOpenMethodsWindow() throws
    DebugError;
    IDebugWindow MenuOpenStaticsWindow() throws
    DebugError;
    IDebugWindow MenuOpenCatchWindow() throws
    DebugError;
    IDebugWindow MenuOpenProcWindow() throws DebugError;
    IDebugWindow MenuOpenOutputWindow() throws
    DebugError;
```

```
    IDebugWindow MenuOpenBreakWindow() throws
    DebugError;
        IDebugWindow MenuOpenLocalsWindow() throws
    DebugError;
        IDebugWindow MenuOpenInspectWindow() throws
    DebugError;
        IDebugWindow MenuOpenRowVarWindow() throws
    DebugError;
        IDebugWindow MenuOpenQueryWindow() throws
    DebugError;
        IDebugWindow MenuOpenEvaluateWindow() throws
    DebugError;
        IDebugWindow MenuOpenGlobalsWindow() throws
    DebugError;
        IDebugWindow MenuOpenConnectionWindow() throws
    DebugError;
        IDebugWindow MenuOpenThreadsWindow() throws
    DebugError;
    IDebugWindow GetWindow( String name ) throws
    DebugError;

        void MenuRunRestart() throws DebugError;
        void MenuRunHome() throws DebugError;
        void MenuRunGo() throws DebugError;
        void MenuRunToCursor() throws DebugError;
        void MenuRunInterrupt() throws DebugError;
        void MenuRunOver() throws DebugError;
        void MenuRunInto() throws DebugError;
        void MenuRunIntoSpecial() throws DebugError;
        void MenuRunOut() throws DebugError;
        void MenuStackUp() throws DebugError;
        void MenuStackDown() throws DebugError;
        void MenuStackBottom() throws DebugError;
        void MenuFileExit() throws DebugError;
        void MenuFileOpen( String name ) throws DebugError;
        void MenuFileAddSourcePath( String what ) throws
    DebugError;
        void MenuSettingsLoadState( String file ) throws
    DebugError;
        void MenuSettingsSaveState( String file ) throws
    DebugError;
        void MenuWindowTile() throws DebugError;
        void MenuWindowCascade() throws DebugError;
        void MenuWindowRefresh() throws DebugError;
        void MenuHelpWindow() throws DebugError;
        void MenuHelpContents() throws DebugError;
```

```
void MenuHelpIndex() throws DebugError;
void MenuHelpAbout() throws DebugError;
void MenuBreakAtCursor() throws DebugError;
void MenuBreakClearAll() throws DebugError;
void MenuBreakEnableAll() throws DebugError;
void MenuBreakDisableAll() throws DebugError;
void MenuSearchFind( IDebugWindow w, String what )
throws DebugError;
void MenuSearchNext( IDebugWindow w ) throws
DebugError;
void MenuSearchPrev( IDebugWindow w ) throws
DebugError;
void MenuConnectionLogin() throws DebugError;
void MenuConnectionReleaseSelected() throws
DebugError;

// output window
void OutputClear();
void OutputLine( String line );
void OutputLineNoUpdate( String line );
void OutputUpdate();

// Java source search path

void SetSourcePath( String path ) throws DebugError;
String GetSourcePath() throws DebugError;

// Catch java exceptions
Vector GetCatching();
void Catch( boolean on, String name ) throws
DebugError;

// Database connections
int ConnectionCount();
void ConnectionRelease( int index );
void ConnectionAttach( int index );
String ConnectionName( int index );
void ConnectionSelect( int index );

// Login to database
boolean LoggedIn();
void Login( String url, String userId, String
password, String userToDebug ) throws DebugError;
void Logout();

// Simulate keyboard/mouse actions
```

```
        void DeleteItemAt( IDebugWindow w, int row ) throws
DebugError;
        void DoubleClickOn( IDebugWindow w, int row ) throws
DebugError;

        // Breakpoints
        Object BreakSet( String where ) throws DebugError;
        void BreakClear( Object b ) throws DebugError;
        void BreakEnable( Object b, boolean enabled ) throws
DebugError;
        void BreakSetCount( Object b, int count ) throws
DebugError;
        int BreakGetCount( Object b ) throws DebugError;
        void BreakSetCondition( Object b, String condition
) throws DebugError;
        String BreakGetCondition( Object b ) throws
DebugError;
        Vector GetBreaks() throws DebugError;

        // Scripting
        void RunScript( String args[] ) throws DebugError;
        void AddEventHandler( DebugScript s );
        void RemoveEventHandler( DebugScript s );

        // Miscellaneous
        void EvalRun( String expr ) throws DebugError;
        void QueryRun( String query ) throws DebugError;
        void QueryMoreRows() throws DebugError;
        Vector GetclassNames();
        Vector GetProcedureNames();
        Vector WindowContents( IDebugWindow window ) throws
DebugError;
        boolean AtBreak();
        boolean IsRunning();
        boolean AtStackTop();
        boolean AtStackBottom();
        void SetStatusText( String msg );
        String GetStatusText();
        void WaitCursor();
        void OldCursor();
        void Error( Exception x );
        void Error( String msg );
        void Warning( String msg );
        String Ask( String title );
        boolean MenuIsChecked( String cmd );
        void MenuSetChecked( String cmd, boolean on );
```

```
void AddInspectItem( String s ) throws DebugError;

    // Constants for DebugScript.OnEvent parameter
    public static final int EventBreak = 0;
    public static final int EventTerminate = 1;
    public static final int EventStep = 2;
    public static final int EventInterrupt = 3;
    public static final int EventException = 4;
    public static final int EventConnect = 5;
}
```

## **sybase.asa.procdebug.IDebugWindow interface**

You can write scripts to control debugger behavior. In scripts, the debugger window is represented by the IDebugWindow interface.

The IDebugWindow interfaces is as follows:

```
// this interface represents a debugger window
package sybase.asa.procdebug;
public interface IDebugWindow
{
    public int GetSelected();
    /*
        get the currently selected row, or -1 if no
    selection
    */

    public boolean SetSelected( int i );
    /*
        set the currently selected row. Ignored if i <
    0 or i > #rows
    */

    public String StringAt( int row );
    /*
        get the String representation of the Nth row of
    the window. Returns null if row > # rows
    */

    public java.awt.Rectangle GetPosition();
    public void SetPosition( java.awt.Rectangle r );
    /*
        get/set the windows position within the frame
    */
```

```
    public void Close();  
    /*  
     * Close (destroy) the window  
     */  
}
```



# Index

## Numerics

64-bit platforms  
    supported server class 729

**A**

Access  
    ODBC configuration for 100  
ad hoc joins  
    performance 285

Adaptive Server Enterprise  
    inserting data from 80, 354  
    inserting text and images 355

administrative scripts 44  
    automatic job scheduling 210  
    generating 208  
    synchronizing query servers 44  
    using 210

aggregates 271

ALL permissions 570

ALLOW\_NULLS\_BY\_DEFAULT option  
    Open Client 687

alphabetic characters  
    defined 538

ALTER DBSPACE statement  
    ADD parameter 230  
    changing read-write mode 232  
    example 237  
    examples 230  
    relocate mode 233  
    RELOCATE parameter 237  
    SIZE parameter 231

ALTER INDEX statement 359

ALTER permissions 570

ALTER statement  
    automatic commit 473

ALTER TABLE statement  
    foreign keys 462

ANSI code pages  
    about 508  
    choosing 526

AppInfo connection parameter 134

asajdbc server class 724

asaodbc server class 728

ASCII  
    conversion on insert 373  
    conversion option 366  
    conversion performance 371  
    data extraction 318  
    data format 311

ASCII character set  
    about 507

asejdbc server class 726

aseodbc server class 729

asiqdemo database 11

AStart connection parameter 137

ASTMP environment variable  
    disk space 128

ASStop connection parameter 137

AT clause  
    CREATE EXISTING TABLE statement 705

atomic compound statements 410

autocommit mode  
    JDBC 799  
    ODBC 100

AUTOINCREMENT  
    default 453  
    negative numbers 454  
    signed data types 454  
    UltraLite applications 454

AUTOMATIC\_TIMESTAMP option  
    Open Client 687

AutoPreCommit connection parameter 136

AutoStart connection parameter 137

AutoStop connection parameter 137

AVG function 271

## B

backslashes  
Windows raw devices 188

backup log  
about 653  
location 653

BACKUP statement 623

BackupEnd system event 743

backups 665  
about 613  
attended 621  
concurrency 475  
concurrency issues 622  
data included in 617  
database 665  
devices 619, 624  
displaying header file 651  
faster 634  
full 657  
increasing memory 659  
incremental 657

Linux 620  
multiplex 614  
NULL 632  
OS-level 665  
performance issues 658  
privileges required 620  
read-only hardware 628  
recovering from errors 630  
responsibilities 658  
scheduling 657  
specifying tape devices on NT 625  
system-level 634  
third party 631  
unattended 621, 652  
virtual 624, 632  
virtual with SAN 634  
wait time 628

base tables 241  
in join indexes 357

batch operations  
multiplex updates 563

batches  
about 395, 407  
control statements 407  
data definition statements 407

SQL statements allowed 437

communication parameters  
Broadcast 158

BCAST communication parameter  
description 158

bcp support 311

BEGIN TRANSACTION statement  
remote data access 713

binary  
data extraction 318

BIT data  
converting 369  
indexes allowed in 269

blanks  
converting to *NULLs* 380

communication parameters  
BroadcastListener 159

connecting  
BroadcastListener 159

firewalls  
BroadcastListener 159

TCP/IP  
BroadcastListener 159

BLISTENER communication parameter  
description 159

BLOB 355

BLOCK FACTOR  
BACKUP statement option 627  
load option 333

block mode 620

BLOCK SIZE  
LOAD TABLE option 334

block size 191

breakpoints  
setting in a Java class 824  
setting in a stored procedure 821

Broadcast communication parameter  
description 158

BroadcastListener communication parameter  
description 159

buffer size  
ODBC configuration 103

build number 23

bulk copy 311

BYTE ORDER option  
LOAD TABLE statement 334

**C**

- c switch 51
- cache
  - writing to 474
- cache size
  - setting for Catalog Store 51
- CALL statement
  - about 396
  - examples 400
  - parameters 413
  - syntax 408
- candidate keys
  - restrictions 446
- case sensitivity
  - collations 537
  - command line 45
  - connection parameters 133
  - database and server names 49
  - international aspects 515
  - passwords 548
  - query results 537
  - remote access 719
- CASE statement
  - syntax 408
- CATALOG ONLY
  - RESTORE option 651
- Catalog Store
  - about 7
  - setting cache size 51
- CBSIZE connection parameter 138
- CDATA
  - in DTD for XML 758
- CDATA section
  - in XML attribute values 784
  - using in XML 782
  - XML markup characters 782
- Certicom
  - obtaining certificates for transport-layer security 606
- certificate authorities
  - transport-layer security 600
- certificate chains
  - transport-layer security 601
- Certificate communication parameter
  - description 159
- certificate fields
  - verifying for transport-layer security 608
- Certificate\_Password communication parameter
  - description 160
- certificates
  - sample certificates 597
- CHAINED option
  - Open Client 687
- chains of certificates
  - transport-layer security 594
- changing
  - collations 547
- CHAR data
  - zero-length cells 374
- character data types
  - matching Adaptive Server Enterprise and Adaptive Server IQ data 384
- character set translation 544
  - error messages 531
- character sets
  - about 503
  - applications 518
  - avoiding translation 533
  - choosing 539
  - connection parameter 138
  - definition 506
  - determining 518
  - encoding 503, 505
  - fixed width 513
  - multibyte 513, 530
  - servers 518
  - single-byte 507
  - translation 544
  - Unicode 530
  - variable width 513
  - Windows 508
  - XML 756, 759
- character text
  - in XML 758
- characters
  - alphabetic 538
  - digits 538
  - white space 538
- CharSet connection parameter 138
- CHECK conditions
  - columns 456
  - deleting 459

- modifying 459
- tables 459
- user-defined data types 458
- checkpoints
  - about 492
  - automatic and explicit 493
  - in recovery 492
  - in system recovery 497
- choosing drivers
  - using the iAnywhere JDBC driver 78
  - using the jConnect JDBC driver 78
- ciphers
  - transport-layer security 590
- CIS (Component Integration Services) 677
- Class.forName method
  - loading jConnect 810
- classes
  - importing 815
- CLASSPATH environment variable
  - jConnect 809
  - setting 796
- client applications
  - persistent tables 241
- client architecture
  - Sybase IQ transport-layer security 592
- client process information 134
- ClientPort communication parameter 160
- CLOB 355
- CLOSE statement
  - procedures 420
- CMP index 276
  - recommended use 276
  - restrictions 277
- code pages
  - ANSI 508
  - definition 506
  - OEM 508
  - overview 507
  - Windows 508
- collation file
  - editing 534
- collations
  - about 503, 514
  - changing 547
  - choosing 539
  - creating 544
- custom 534, 544, 546
- definition 506
- file format 534
- internals 534
- ISO\_1 527
- multibyte 530
- OEM 528
- recommended for Japanese 530
- WIN\_LATIN1 527
- collisions
  - dynamic 212
  - static 212
- column delimiters
  - load format option 329
  - LOAD TABLE statement 327
- column names
  - international aspects 515
- column set to during load 377
- column width
  - insertion issues 370
- columns
  - adding 245
  - changing 245
  - defaults 447
  - deleting 245
  - properties 451
  - retrieving row identifiers 15
- command delimiter
  - setting 435
- command files
  - creating database objects 178
  - DBISQL 177
- command-line switches 45
  - displaying 45
  - required 46
- CommBufferSize connection parameter 138
- COMMIT statement
  - compound statements 410
  - JDBC 799
  - procedures 434
  - remote data access 713
- committing transactions
  - effect of timing on read transactions 482
  - in DBISQL 473
- CommLinks connection parameter 139
- communication parameter 159

communication parameters  
 about 133  
 Certificate 159  
 Certificate\_Password 160  
 ClientPort 160  
 description 168  
 DLL 162  
 DOBROADCAST 161  
 HOST 163  
 MaxConnections 168  
 MYIP 168  
 PreFetchOnOpen 169  
 ServerPort 170  
 SESSIONS 171  
 TDS 172  
 TIMEOUT 172, 173  
 communications  
 parameters 157  
 Compare index  
*See CMP index*  
 compound statements  
 atomic 410  
 declarations 409  
 using 409  
 concurrency  
 backups 475, 622  
 data definition 486  
 in Adaptive Server IQ 474  
 insertions, deletions, and queries 486  
 read and write 478  
 configuration  
 preserving 208  
 configuration files  
 using 45  
 configuring  
 ODBC data sources 99  
 connect  
 permission 567  
 CONNECT statement  
 default database 117  
 ConnectFailed event handler 559  
 connecting  
 character sets 532  
 firewalls 160  
 jConnect 812  
 LDAP communication parameter 164  
 connection handle 24, 347  
 connection name  
 ODBC configuration 104  
 connection parameter 147  
 connection parameters  
 about 133  
 case insensitivity 133  
 CharSet 138  
 conflicts 109  
 data sources 95  
 default 94  
 embedded databases 109  
 in connection strings 74  
 location of 112  
 priority 134  
 connection profiles 83  
 connection strings  
 character sets 532  
 representing 74  
 connection system events 743  
 ConnectionName connection parameter 141  
 connections  
 debugging 819  
 default database 117  
 dropping 127  
 embedded database 91  
 establishing 74  
 examples 85  
 how the server establishes 110  
 Interactive SQL 86, 117  
 jConnect URL 811  
 JDBC 73, 792  
 JDBC client 793  
 JDBC defaults 800  
 JDBC example 797  
 JDBC examples 793  
 JDBC in the server 797  
 limiting concurrent 50  
 local database 86  
 logging 127  
 managing 556  
 overview 72  
 remote 78, 713  
 restricting 639  
 to database on foreign host 90  
 troubleshooting 127

using data source 93  
connectivity  
  iAnywhere JDBC driver 78  
  jConnect 78  
constant expression defaults 455  
constraints  
  effect on performance 445  
Containment index  
  *See* WD index  
CONTINUE\_AFTER\_RAISERROR option  
  Open Client 687  
control statements  
  list 408  
conventions  
  documentation xxvii, xxviii  
  syntax xxvii  
  typographic xxviii  
conversion options  
  DATE 374  
  DATE format specification 375  
  DATETIME 377, 378  
  flat file loads 366  
  performance 371  
  substitution for zero-length cells 374  
CONVERSION\_ERROR database option 387  
conversions  
  between Adaptive Server Enterprise and Adaptive Server IQ 382  
  errors on import 387  
  insert options 366  
  on insert 364  
converting query servers to write servers 666  
COUNT DISTINCT  
  impact on index choice 265  
COUNT function 271  
CREATE DATABASE statement 186  
  IQ RESERVE parameter 228  
  IQ SIZE parameter 228  
  TEMPORARY RESERVE parameter 228  
  TEMPORARY SIZE parameter 228  
CREATE DBSPACE statement 216  
  RESERVE parameter 228  
  reusing space 236  
  SIZE parameter 228, 236  
CREATE EXISTING TABLE statement  
  using 706  
CREATE INDEX statement 261  
CREATE JOIN INDEX statement 243, 299  
CREATE PROCEDURE statement  
  examples 398  
  parameters 412  
Create Query Server wizard 194  
CREATE statement  
  automatic commit 473  
  concurrency rules 486  
CREATE TABLE statement  
  and command files 178  
  example 239  
  proxy tables 707  
creating  
  column defaults 448  
creating query servers 194  
creating the certificates  
  transport-layer security 603  
cryptography  
  Sybase IQ public key 591  
CS connection parameter 138  
CS\_TEXT\_TYPE 355  
CT-library  
  about 677  
curly braces 355  
current date and time defaults 452  
cursors  
  and LOOP statement 422  
  command syntax 502  
  connection limit 584  
  examples 502  
  hold 483, 501  
  in procedures 422  
  in transactions 500  
  message logging 502  
  ODBC configuration 100  
  on SELECT statements 422  
  positioned operations 502  
  procedures 420  
  scrolling 501  
  sensitivity 501  
custom collations  
  about 534  
  creating 534  
  creating databases 546

**D**

## data

- deleting 393
- duplicated 444
- exporting 309, 313
- extracting 315
- importing 309
- in transactions 481
- input and output formats 311
- invalid 444
- loading 309

## data definition

- concurrency rules 486

## data definition language

- about 175

## data definitions

- creating 239

## data extraction

- about 315
- ASCII 318
- binary 318
- binary/swap 318
- controlling access 318
- options 315
- options list 315

## data integrity

- column defaults 447
- constraints 447
- overview 443
- rules in the system tables 469

## data modification

- permissions 312

## data replication

- 476

## data source description

- ODBC 99

## data source name

- ODBC 99

## data sources

- about 95
- command line creation 98
- configuring 99
- connecting with 93
- external servers 727
- ODBC 95
- UNIX 105
- using with jConnect 78

## data types

- character 384
- conversion during loading 366
- converting 364
- converting between Adaptive Server Enterprise and Sybase IQ 382
- creating with **sp\_addtype** 14
- dropping user-defined 14
- FLOAT* 384
- integer 383
- matching Sybase IQ and Adaptive Server Enterprise 382
- money 385, 386
- REAL* 384
- retrieving 15
- specifying in table creation 240

## database administrator

- defined 550
- See Also* DBA 550

## database file

- ODBC configuration 102

## database name

- ODBC configuration 101

## database options

- changing or displaying 14
- Open Client 687
- startup settings for TDS connections 688

## database server

- about 6
- as Windows service 45
- command-line switches 45
- connecting 78
- emergency stop 131
- name caching 116
- name switch 47
- naming at startup 48
- remote 78
- starting 31
- starting at command prompt 33
- starting from Start menu 44
- starting on Windows 44
- stopping 62, 66

## database servers

- preventing from starting 137
- stopping 156

## DatabaseFile connection parameter 141

- DatabaseName communication parameter
    - description 161
  - DatabaseName connection parameter 143
  - databases
    - activating multiplex capability 194
    - Adaptive Server Anywhere 7
    - Adaptive Server IQ data 7
    - backup 665
    - block size 191
    - character set 531
    - checking consistency 637
    - choosing a location 187
    - connecting to 72, 78, 110
    - creating 183
    - creating objects 239
    - creating with utility database 564
    - custom collations 546
    - DBCC consistency checker 637
    - default characteristics 186
    - designing 176
    - displaying status information 13
    - displaying validation results 12
    - dropping 216
    - Embedded 134, 138
    - estimating space requirements 12
    - file location 185
    - initializing 183
    - listing size 11
    - management tasks 3
    - managing with Interactive SQL 177
    - moving 185
    - moving files 643
    - multiple 709
    - multiple on server 690
    - naming 47
    - naming at startup 48
    - overview of setup 178
    - owner role 4
    - page size 189
    - permission to create and drop 565
    - permission to start 67
    - permissions 5, 53, 549
    - preallocating space 180
    - privileges needed to create 179
    - proxy 677
    - relative pathnames 187
  - sample xxix
  - schema creation 239
  - security overview 4
  - size 189
  - stopping 67
  - temporary data 7
  - unloading 66
  - URL 811
  - utility 26
  - validating 11, 637
  - very large 26
  - working with objects 175
- DatabaseStart system event 743
- DatabaseSwitches connection parameter 144
- DataSourceName connection parameter 144
- DATE data type
  - optimizing loads 371
  - specifying format for conversion 375
- date data types
  - matching Adaptive Server Enterprise and Adaptive Server IQ data 385
- DATE format
  - converting two-digit dates 376
- DATE index 279
  - additional indexes 284
  - advantages 284
  - comparison to other indexes 284
  - disadvantages 284
  - recommended use 280
- DATE option 366, 374
- DATEPART
  - queries 280
- dates
  - procedures 436
- DATETIME
  - conversion option 366
  - load conversion option 377
- DATETIME data type 377
  - format for conversion 378
  - optimizing loads 371
- Datetime index
  - See DTTM index*
- DBA (database administrator)
  - defined 550
  - responsibilities 3
  - role of 4

DBA authority  
     about 550  
     granting 551, 569  
     not inheritable 574

DBA user 569  
     dropping 569

DBASE formats 311

DBCC\_LOG\_PROGRESS option 638

DBCOLLAT utility 546  
     custom collations 544

DBF connection parameter 141  
     embedded databases 91

DBISQL  
     command line parameters 89  
     committing transactions 473  
     inserting data interactively 357  
     introduction 177  
     logon window 90  
     *See Also* Interactive SQL 177  
     specifying output format 312

DBKEY connection parameter 145

DB-Library  
     about 677

DBLOCATE utility 130

dblog utility 649

communication parameters  
     DatabaseName 161

DBN communication parameter  
     description 161

DBN connection parameter 143

dbo user ID  
     views owned by 252

DBS connection parameter 144

dbspace  
     adding local 221  
     adding temporary 221  
     changing the read-write mode 237  
     changing the size 230  
     create example 228, 236  
     creating 216  
     definition 180  
     displaying index information 12, 229  
     displaying usage information 11, 229  
     dropping 224, 227, 235, 236, 238  
     dropping local 226  
     dropping temporary 227

estimating space requirements 12

file location when creating 185

management example 228

multiplex 221

read-write mode 232

relocate mode 232

relocating objects 13, 233

restoring to raw device 644

reusing space 236

dbspace read-only mode 232

DDL  
     about 175

DDL (Data Definition Language) 22

DDL locking 486

DDL statements  
     collisions 212

debugger  
     about 817  
     connecting 819  
     getting started 819  
     requirements 818  
     starting 819  
     tutorial 819

debugger features 817

debugger utility  
     features 817

debugging  
     breakpoints 824  
     compiling classes 823  
     connection 819  
     event handlers 750  
     features 817  
     introduction 817  
     Java 823  
     local variables 822  
     permissions 818  
     requirements 818  
     stored procedures 820

DebugScript class 829

DECLARE statement  
     compound statements 409  
     procedures 420, 426

default configuration file 45

default index  
     about 270

defaults

- AUTOINCREMENT 453
  - column 447
  - connection parameters 94
  - constant expressions 455
  - creating 448
  - creating in Sybase Central 451
  - current date and time 452
  - inserting 449
  - loading 449
  - NULL 454
  - string and number 454
  - USER special value 453
- DELETE permissions 570
  - deleting
    - column defaults 449
    - deleting query servers 208, 211
- DELIMITED BY option 329
- delimiters
  - SELECT statement 355
- Delphi
  - ODBC configuration for 100
- device types
  - for databases 180
- digit characters
  - defined 538
- digital certificates
  - transport-layer security 592
- Directory Services Editor
  - See DSEdit*
- DisableMultiRowFetch connection parameter 145
- disconnecting 126
- disk resources
  - losing 669
- disk space
  - allocating 216
  - indexes 268
  - saving 501
- disk storage 665
- distributed applications
  - about 813
  - example 815
  - requirements 813
- DLL communication parameter 162
- DML (Data Manipulation Language) 22
- DMRF connection parameter 145
- DOBROADCAST communication parameter 161
- Document Type Definition
  - See DTD*
- documentation
  - accessibility features xxix
  - Adaptive Server Anywhere xxv
  - Adaptive Server IQ xxiv
  - conventions xxvii, xxviii
  - on CD xxvi
  - online xxvi
- DOM
  - Document Object Module XML parser 762
  - See also XML parsers*
- Driver Not Capable error
  - ODBC configuration 100
- driver, missing 719
- drivers
  - iAnywhere JDBC driver 78
  - jConnect JDBC driver 78
- DROP DBSPACE statement
  - dropping a main dbspace 235
  - dropping a temp dbspace 236, 238
- DROP DOMAIN statement
  - query servers 207
- DROP JOIN INDEX statement 243
- DROP statement
  - automatic commit 473
  - concurrency rules 486
- DROP TABLE statement
  - example 246
- DROP VIEW statement
  - example 253
  - restriction 252
- dropping 569
  - dbspaces 226, 227
  - last local dbspace on query server 226
  - main dbspace on multiplex 227
  - query servers 208
  - views 252
- dropping connections 127
- DSEdit 80
  - entries 682
  - starting 680
  - using 680
- DSN connection parameter 144
  - about 95
- DSS (decision support system) 2

- DTD
    - elements of 758
    - embedded 758
    - embedding in XML document 757
    - external 758
    - internal 758
    - use with XML 757
  - DTTM index 279
    - additional indexes 284
    - advantages 284
    - comparison to other indexes 284
    - disadvantages 284
    - recommended use 280
  - dynamic collisions 212
- 
- ## E
- Embedded databases 134, 138
  - embedded databases
    - connecting 91
    - connection parameters 109
    - Java 92
    - starting 91
  - client side
    - Encryption 147
  - communications
    - Encryption 147
  - connection parameters
    - Encryption 147
  - encryption
    - Encryption 147
  - security
    - Encryption 147
    - strong encryption
      - Encryption 147
  - ENC connection parameter
    - description 147
  - encoding
    - character sets 506
    - definition 506
    - multibyte character sets 538
  - encrypted passwords
    - ODBC configuration 101
  - EncryptedPassword connection parameter 146
  - encryption 590
- communications 172
  - hiding objects 436
  - strong 147
- Encryption connection parameter
    - description 147
  - ENG connection parameter 146
  - EngineName connection parameter 146
  - ENP connection parameter 146
  - enterprise root certificates
    - creating for transport-layer security 603
    - transport-layer security 602
  - entity integrity
    - about 446
    - enforcing 460
  - environment variables
    - SQLCONNECT 95
  - error handling
    - ON EXCEPTION RESUME 427
  - error messages 208
    - character set translation 531
    - PIPE\_NOT\_CONNECTED 329
    - redirecting to files 314
  - error system events 743
  - errors
    - data conversion 387
    - insertions and deletions 486
    - procedures 424
    - transaction processing 486
  - ESCAPE CHARACTER option
    - LOAD TABLE option 333
  - euro symbol
    - 1252LATIN1 collation 527
  - event handlers 745
    - ConnectFailed 559
    - debugging 750
    - triggering 750
  - event type
    - active 747
    - polled 747
  - events 739–751
    - adding 749
    - retrieving a schedule name 751
    - retrieving an event name 751
    - start time 742
    - system 743
    - trigger condition 742

## *Index*

Excel format 311  
exception handlers  
  procedures 430  
exceptions  
  declaring 426  
excluding servers 207  
**EXECUTE IMMEDIATE** statement  
  procedures 434  
**executeQuery** method  
  about 804  
**executeUpdate** method  
  about 801  
executing DDL commands 239  
exporting data  
  about 313  
  overview 309  
extended characters  
  about 507  
Extensible Markup Language  
  *See XML*  
Extensible Style Language  
  *See XSL*  
external logins  
  about 703  
  creating 703  
  dropping 704  
extracting data  
  about 315  
  options 315  
  options list 315

**F**

failover 669  
failures  
  media 613  
  system 613  
Federal Rehabilitation Act  
  section 508 xxix  
fetch operation  
  suppressing warnings 100  
**FETCH** statement  
  procedures 420  
FileDataSourceName connection parameter 149  
FileDSN

connection parameter 95  
creating 104  
distributing 105  
files  
  redirecting output to 313, 315  
**FILLER** column  
  maximum length 328  
**FILLER** option 363  
FIPS  
  about 590  
FIPS 140-2 certification  
  about 590  
firewalls  
  connecting across 160  
  LDAP communication parameter 164  
**FIXED** format 311  
fixed width character sets  
  about 513  
flat files  
  load conversion options 366  
  loading from 324  
**FLOAT\_AS\_DOUBLE** option  
  Open Client 687  
follow bytes  
  about 514  
**FOR** statement  
  syntax 408  
forced recovery  
  multiplex 668  
foreign keys  
  creating 248  
  existing unenforced 465  
  inserting data 358  
  optional 465  
  restrictions 446  
  retrieving information 15  
  unenforced 248  
**FORWARD TO** statement 710  
FoxPro format 311  
**FROM** clause  
  join indexes 288  
  `UPDATE` statement 392  
functions  
  `BFILE` 355  
  for `BLOB` 355  
  for `CLOB` 355

types of 22  
 user-defined 404

threads 720

**G**

getConnection method  
     instances 800

global certificates  
     using as a server certificate for transport-layer security 607

global temporary tables  
     about 241

globally signed certificates  
     transport-layer security 605

gm switch 50  
     effect on recovery 496

GRANT statement  
     creating groups 575  
     DBA authority 569  
     group membership 576  
     JDBC 808  
     new users 567  
     passwords 568  
     permissions 570  
     procedures 572  
     RESOURCE authority 569  
     WITH GRANT OPTION 572  
     without password 577

GROUP BY clause  
     impact on index choice 265

GROUP permissions  
     not inheritable 574

groups  
     adding with **sp\_addgroup** 14  
     changing membership 14  
     creating 575  
     dropping 14  
     managing 574  
     membership 576  
     permissions 553, 576  
     PUBLIC 578  
     Sybase Central 575  
     SYS 578  
     without passwords 577

-gx option

**H**

HG index  
     additional indexes 274  
     advantages 273  
     automatic creation 274  
     comparison to other indexes 274  
     disadvantages 273  
     foreign key constraint 273  
     multicolumn with NULL 275  
     NULL values 275  
     query performance 275  
     recommended use 273

High\_Group index  
     *See HG index*

High\_Non\_Group index  
     *See HNG index*

HNG index 275  
     additional indexes 276  
     advantages 276  
     comparison to other indexes 276  
     disadvantages 276  
     recommended use 275

hold cursors 483, 501

HOST communication parameter 163

HTML  
     comparison to XML 753

Hypertext Markup Language  
     *See HTML*

hyperthreading  
     server switch 52

**I**

IANA  
     port number 170

iAnywhere JDBC driver  
     choosing a JDBC driver 788  
     connecting to ASA databases 78

IDebugAPI interface 830

IDebugWindow 834

identifiers

- case insensitivity 515
- international aspects 515
- connection parameters
  - Idle 149
- Idle connection parameter
  - description 149
- IF statement
  - syntax 408
- import
  - jConnect 810
- importing data
  - conversion errors 387
  - from Adaptive Server Enterprise 354
  - from pre-Version 12 IQ databases 354
  - LOAD TABLE statement 324
- in LOAD TABLE 377
- including servers 207
- index types
  - about 258
  - criteria for choosing 264
  - LF 271
  - recommendations 266
  - selecting 285
- indexes
  - about 257
  - adding after loading tables 286
  - adding and dropping 260
  - created automatically 243
  - creating 239, 261
  - creating in Sybase Central 263
  - disk space usage 268
  - displaying size 12
  - dropping 256
  - in system tables 255
  - introduction 254
  - listing 12
  - multicolumn HG and NULL 275
  - parallel creation 263
  - rebuilding 245
  - renaming 256
  - selecting an index type 285
  - validating 256
- insert conversion options 366
- INSERT LOCATION statement 354
- INSERT permissions 570
- INSERT statement 287
- about 351
- and integrity 445
- incremental 353
- JDBC 801, 803
- objects 807, 808
- partial-width insert 359
- performance 353
- VALUES option 352
- inserting
  - column defaults 449
  - column width issues 370
  - from Adaptive Server Enterprise database 354
  - from older versions 366
  - from other databases 354
  - interactively 357
  - join index tables 357
  - overview 309
  - partial-width inserts 359
  - performance 387
  - primary and foreign key columns 358
- See Also* loading data 366
- selected rows 353
- integer data types
  - matching Adaptive Server Enterprise and Adaptive Server IQ 383
- Integrated connection parameter 150
- integrated logins
  - default user 125
  - network aspects 125
  - ODBC configuration 101
  - operating systems 118
  - using 122
- integrity
  - column defaults 447
  - constraints 446, 447
  - overview 443
- Interactive SQL
  - command delimiter 435
- See Also* DBISQL 177
- window problems 132
- interface libraries
  - connections 72
- interfaces
  - IDebugAPI 830
  - IDebugWindow 834
- interfaces file

adding entries 80  
 configuring 679  
 internal build number 23  
 INTO clause  
     using 415  
 invoking transport-layer security 596  
 IP address 134  
     about 682  
 IP communication parameter 163  
 IPX  
     server configuration 157  
**IQ Agent**  
     configuring 35  
     host alias 36  
     host name 36  
     log file 38  
     overriding port number 37  
     owner 37  
     port number 36  
     purpose 35  
     running 35  
**IQ PAGE SIZE** 189  
**IQ server**  
     troubleshooting startup 597  
**IQ Store**  
     content and structure 7  
**IQ UNIQUE**  
     changing value of 245  
     performance impact 244  
**IQ UNIQUE constraint** 456  
**IQ UNIQUE table option** 243  
**IQ User Administration**  
     user permissions 126  
**iqdsn command** 98  
**iqgovern switch** 51  
**iqnumbercplus**  
     server switch 52  
**iqwmem switch** 51  
**ISO\_1 collation**  
     about 527  
**Isolation level**  
     ODBC configuration 99  
 isolation levels 490  
**ISOLATION\_LEVEL option**  
     Open Client 687  
**ISQL**

connections 80

## J

**Java**  
     about 817  
     about debugging 817  
     debugging 817, 823  
     JDBC 787  
     memory requirements 51  
     querying objects 813  
     use in Adaptive Server IQ 73, 191  
**Java data types**  
     inserting 807  
     retrieving 807  
**Java debugger**  
     requirements 818  
     starting 819  
     tutorial 819  
**jcatalog.sql file**  
     jConnect 810  
**jConnect**  
     about 808  
     choosing a JDBC driver 788  
     CLASSPATH environment variable 809  
     connections 793, 797  
     database setup 810  
     installation 809  
     jdbcdrv.zip 809  
     loading 810  
     packages 810  
     system objects 810  
     TDS 679  
     URL 811  
     versions 809  
**jConnect driver**  
     connecting to IQ databases 78  
**JDBC**  
     about 787  
     applications overview 789  
     autocommit 799  
     client connections 793  
     client-side 792  
     configuring connections 80  
     connecting 792

- connecting to a database 811
- connection code 793
- connection defaults 800
- connections 73, 792
- data access 800
- examples 793
- features 790
- INSERT statement 801, 803
- jConnect 808
- non-standard classes 790
- overview 787
- permissions 808
- prepared statements 806
- requirements 788
- SELECT statement 804
- server-side 792
- server-side connections 797
- version 790
- ways to use 787
- JDBC connectivity
  - about 78
- JDBC drivers
  - choosing 788
  - compatibility 788
  - performance 788
- JDBCExamples class
  - about 801
- join columns 296
- join hierarchy 288
- join indexes
  - about 287
  - altering columns 246
  - base tables 357
  - candidate keys 446
  - columns in tables 289
  - creating 239, 293
  - creating in Sybase Central 301
  - estimating benefit 308
  - estimating size 307
  - estimating space requirements 12
  - inserting into 357
  - join hierarchy 288
  - join relationships 296
  - listing size 12
  - modifying underlying tables 305
  - on multiplex 243
- ratio of top to related table rows 308
- synchronizing 295
- join relationships
  - defining 296
  - specifying 298
- join types 301
- join virtual tables 243
- joins
  - linear 302
  - multi-table 292
  - performance impact 285
  - star 302
  - updates using 392

## **K**

- key joins 296, 300
- keyboard mapping
  - about 506
- keywords
  - remote servers 718

## **L**

- connection parameters
  - Language 151
- LANG connection parameter
  - description 151
- language
  - locale 517
- Language connection parameter
  - description 151
- language resource library
  - messages file 506
- language support
  - about 503
  - collations 539
  - multibyte character sets 530
  - overview 503
- Large Objects Management 355
- LazyClose connection parameter
  - description 151
- connection parameters
  - LazyClose 151

LCLOSE connection parameter  
     description 151

communication parameters  
     LDAP 164

LDAP communication parameter  
     description 164

LDAP servers  
     LDAP communication parameter 164

LEAVE statement  
     syntax 408

communication parameters  
     LogFormat 165

LF communication parameter  
     description 165

LF index 271  
     additional indexes 272  
     advantages 272  
     comparison to other indexes 272  
     disadvantages 272  
     recommended use 271

libctl.cfg file  
     DSEDITION 681

LIMIT option  
     LOAD TABLE statement 335

Links connection parameter 139

liveness  
     ODBC configuration 103

LivenessTimeout connection parameter 152

load conversions  
     *See* conversion options

load optimization 371

load options 332

load performance  
     lock contention 489

LOAD TABLE statement 287  
     about 324

FILLER option 363

IGNORE CONSTRAINT option 338, 339

integrity constraints 325

LOG DELIMITED BY 342

MESSAGE LOG option 341

partial-width insert 359

performance 371

QUOTES option 330

QUOTES option example 330

ROW LOG 341

syntax 325

WORD SKIP option 332

loading  
     column defaults 449  
     jConnect 810

loading data  
     ASCII conversion option 373  
     ASE data 355  
     concurrency rules 486  
     conversion errors 387  
     conversion options 366  
     file specification 328  
     format options 329  
     integrity constraint violations 347  
     large objects 355  
     logging constraint violations 347  
     named pipes 329  
     notification messages 343  
     overview 309  
     performance 387  
     persistent tables 241  
     privileges needed 312  
     *See Also* inserting 366

loads  
     thread usage 391

LOB 355

communication parameters  
     LocalOnly 164

LOCAL communication parameter  
     description 164

local temporary tables  
     about 241

locale  
     character sets 531  
     language 517

locales  
     about 516  
     setting 541

localhost  
     machine name 682

LocalOnly communication parameter  
     description 164

lock contention  
     managing 489

locking  
     DDL operations 486

- tables 485
- lockout
  - automatic 559
- communication parameters
  - LogFile 165
- LOG communication parameter
  - description 165
- Log file
  - server 60
- LogFile communication parameter
  - description 165
- LogFile connection parameter 153
- LogFormat communication parameter
  - description 165
- logging connections 127
- login failures 559
- Login Management
  - collisions 563
  - multiplex 561
- Login Management
  - batch operations 563
- LOGIN\_MODE database option
  - integrated logins 120
- logins
  - customizing for query server 205
  - integrated 118, 119
  - limiting 556
- LogMaxSize communication parameter
  - description 166
- LogOptions communication parameter
  - description 167
- LOOP statement
  - in procedures 422
  - syntax 408
- communication parameters
  - LogOptions 167
- LOPT communication parameter
  - description 167
- Lotus format 311
- Low\_Fast index
  - See* LF index
- lower code page
  - about 507
- communication parameters
  - LogMaxSize 166
- LSIZE communication parameter
  - description 166
- LTO connection parameter 152

## M

- main dbspace, adding 219
- making a new self-signed certificate
  - transport-layer security 598
- Managing 549
- managing
  - transactions 714
- MAXCONN communication parameter
  - description 168
- MaxConnections communication parameter
  - description 168
- MaxRequestSize communication parameter
  - description 168
- MAXSIZE communication parameter
  - description 168
- memory
  - connection limit 584
  - creating wired memory pool 51
  - for Catalog Store cache 51
- memory message
  - load notification messages 343
- Message Agent 477
- MESSAGE LOG
  - contents 348
  - example 350
  - LOAD TABLE option 341
  - LOG DELIMITED BY option 342
- message log 23
- MESSAGE statement
  - procedures 426
- messages
  - dropping 14
  - language resource library 506
  - memory notification 343
  - recorded in message log 23
  - redirecting to files 314
  - retrieving stored strings 15
- metadata
  - in Catalog Store 7
- Microsoft Access
  - ODBC configuration for 100

Microsoft Visual Basic

- ODBC configuration for   100

- migration   666

- mixed-version multiplex   203

- modifying

  - column defaults   449

  - modifying and deleting column defaults   449

  - money data types   385, 386

  - moving write server to a new machine   666

  - multibyte character sets

    - about   513

    - using   530

  - multibyte characters

    - encodings   538

    - properties   538

  - multiple databases

    - DSEdit entries   682

    - joins   709

  - multiple record fetching

    - ODBC configuration   104

  - multiplex

    - creating join indexes   243

    - dropping domains   207

    - dropping local dbspace   226

    - login management   561

    - mixed-version   203

    - moving   662

  - multiplex databases

    - adding dbspaces   221

    - backups   614

    - creating tables   239

    - moving to new machine   666

    - restoring   637

    - static collisions   212

    - validating   638

  - multiplex servers

    - balancing loads   173

    - batch operations   563

    - moving files to new machine   666

    - remote data access   712

    - starting   41

    - stopping   62

    - synchronizing   204

    - troubleshooting   208

  - multiprocessor machines

    - switches   49

multithreading

- during loads   391

MYIP communication parameter   168

## N

**-n** switch   35

named pipes   329

- extraction   324

national language support

  - about   503

  - collations   539

  - multibyte character sets   530

  - overview   503, 505

natural joins   300

NEAREST\_CENTURY option   376

NetBIOS

  - server configuration   157

network communications

    - command line switches   157

    - troubleshooting startup   128, 129

network number

    - in IPX address   161

network protocols

    - ODBC configuration   102

NOT NULL constraint   446

notification messages   343

notify count   262

NOTIFY option

    - INSERT statement   289

    - LOAD TABLE statement   335

NULL   377

    - conversion option   366, 380

    - converting to   380

    - default   454

    - inserting   352

    - on multicolumn HG index   275

    - result of partial-width insertions   356

NULL backups   632

NULL value

    - in multicolumn HG index   275

    - output   314

NULLS option

    - DBISQL   314

## O

objects  
    hiding 436  
    inserting 807  
    qualified names 579  
    querying 813  
    retrieving 807, 813  
obtaining server-authentication certificates  
    transport-layer security 606  
ODBC  
    configuring data sources 99  
    connection parameters 133  
    data sources 95  
    driver location 111  
    external servers 727  
    initialization file for UNIX 105  
    UNIX support 105  
ODBC connectivity  
    about 78  
ODBC data sources  
    using with jConnect 78  
ODBC server classes 727  
ODBC translation driver  
    ODBC configuration 99  
OEM code pages  
    about 508  
    choosing 526  
offline status 211  
OmniConnect 677  
    support 685  
ON clause joins 300  
ON EXCEPTION RESUME clause  
    about 427  
    not with exception handling 431  
ON FILE ERROR option  
    LOAD TABLE statement 335  
on-line status 211  
Open Client  
    configuring 679  
    interface 677  
Open Database Connectivity. *See* ODBC  
Open Server  
    adding 679  
    addresses 682  
    architecture 677  
    starting 686  
    system requirements 685  
OPEN statement  
    procedures 420  
operator  
    tasks of 658  
option value  
    truncation 316  
options  
    customizing for query server 205  
    DBCC\_LOG\_PROGRESS 638  
    Open Client 687  
    setting 585  
    startup settings for TDS connections 688  
OrderXml class 763  
OS-level backups 665  
output format  
    DBISQL 312  
output redirection 313, 315  
OUTPUT\_FORMAT  
    DBISQL option 312  
owner  
    role of 4  
owners  
    about 552

## P

packages  
    jConnect 810  
page size 189  
    Catalog 55  
    switch 55  
parallel CREATE INDEX 263  
params.cfg file 45  
parsers for XML 762  
partial-width insertions  
    about 359  
    examples 361  
    rules 360  
    START ROW ID option 356  
password  
    utility database 564  
Password connection parameter 153  
passwords  
    batch operations 563

case sensitivity 548  
 changing 15, 568  
 default 550  
 minimum length 568  
 ODBC configuration 101  
 rules 568  
 setting expiration 556  
 verifying 568  
**pathnames**  
 for databases 187  
**performance**  
 ad hoc joins 285  
 effect of constraints 445  
 impact of versioning 498  
 indexes 254  
 inserts 353  
 JDBC 806  
 JDBC drivers 788  
 loading data 387  
 loading from flat files 367  
 procedures 396  
**permissions**  
 command-line switches 53  
 conflicts 584  
 connect 567  
 customizing for query server 205  
 DBA authority 550  
 debugging 818  
 granting passwords 567  
 group 574  
 group membership 576  
 groups 553, 576  
 in Sybase Central 571, 573  
 individual 566  
 inheriting 572, 574  
 INSERT and DELETE, on views 583  
 JDBC 808  
 listing 586  
 managing 549  
 overview 549  
 passwords 568  
 procedures 402, 572  
 RESOURCE authority 552, 569  
 tables 552, 570  
 the right to grant 572  
 types of 5  
 user-defined functions 406  
 views 252, 570  
 WITH GRANT OPTION 572  
**persistent objects** 206, 241  
 renaming 212  
**ping**  
 testing Open Client 684  
**PIPE\_NOT\_CONNECTED error** 329  
**pipes**  
 named 324  
**plug-ins**  
 connecting 69  
**PORT communication parameter** 170  
**port number**  
 default 90  
 for the database server 170  
 specifying on UNIX 90  
 TCP/IP 686  
**Port option**  
 introduction 686  
**PreFetchOnOpen communication parameter**  
 description 169  
**PREPARE statement**  
 remote data access 713  
**prepared statements**  
 JDBC 806  
**PreparedStatement interface**  
 about 806  
**PREVIEW option**  
 LOAD TABLE statement 336  
**primary keys**  
 AUTOINCREMENT 453  
 creating 247  
 entity integrity 460  
 inserting data 358  
 multicolumn 294, 461  
 order of columns 248  
 retrieving information 15  
 unenforced multi-column 248  
**privileges**  
 defining database objects 179  
 for inserting and deleting 312  
 multiplex login management 562  
**Procedure Not Found error**  
 Java methods 804  
**procedures**

- about 395
- benefits of 396
- calling 400
- command delimiter 435
- creating 398
- cursors 420
- cursors in 422
- dates and times 436
- debugging 820
- default error handling 425
- displaying information about 397
- dropping 401
- error handling 424
- exception handlers 430
- EXECUTE IMMEDIATE statement 434
- execution permissions 402
- multiple result sets from 418
- owner 397, 552
- parameters 397, 412, 413
- performance 396
- permissions 572
- permissions for creating 552
- result sets 403, 417
- returning results 414, 415
- returning results from 402
- savepoints in 434
- security 581
- See Also* stored 10
- SQL statements allowed in 411
- structure 411
- system 10
- table names 435
- using 397
- variable result sets from 419
- verifying input 436
- warnings 429
- writing 434
- profiles
  - connection 83
- properties
  - server 211
- protocols
  - switch 56
- proxy databases 677
- proxy tables
  - about 695, 704
  - creating 695, 704, 706, 707
  - location 705
  - properties 706
- PUBLIC group 578
- public key cryptography
  - Sybase IQ 591
- public-key cryptography
  - Sybase IQ 591
- PWD connection parameter 153

## **Q**

- qualified object names 579
- queries
  - concurrency rules 486
  - JDBC 804
  - limiting concurrent 51
  - performance 275
  - range predicates 282
  - with DATEPART 280
- query server
  - converting to write server 666
  - creating 194
  - customizing options 205
  - definition 9
  - deleting 208, 211
  - dropping 208
  - moving 662
  - persistent tables 241
  - remote data access 712
- query servers
  - DROP DOMAIN 207
  - synchronizing without Sybase Central 44
- query types
  - index types for 265
- questions
  - character sets 505
- quotation marks
  - in SQL 579
- QUOTED\_IDENTIFIER option
  - Open Client 687

**R**

- raw devices 180
  - consistent naming 221
  - naming on Windows 188
  - restoring to 644
  - symbolic links 221
  - utility database 564
- communication parameters
  - ReceiveBufferSize 170
- RCVBUFSZ communication parameter
  - description 170
- read-only hardware
  - backups 628
  - example 674
- REAL* data type
  - matching Adaptive Server Enterprise and Adaptive Server IQ data 384
- ReceiveBufferSize communication parameter
  - description 170
- recovering space 476
- recovery
  - system 496
  - transaction log in 497
  - transactions in 496
- recycling the server 62
- redirecting
  - output to files 313, 315
- REFERENCES permissions 570
- referential integrity
  - column defaults 447
  - declaring 462
  - enforcing 460
  - enforcing with existing unenforced foreign keys 465
  - permissions 312
- RELEASE SAVEPOINT statement 493
- remote data
  - location 705
- remote data access 677
  - case sensitivity 719
  - internal operations 714
  - multiplex servers 712
  - passthrough mode 710
  - proxy tables 354
  - remote servers 696
- SQL Remote unsupported 718
- troubleshooting 718
- unsupported features 718
- remote procedure calls
  - about 711
- remote servers
  - about 696
  - altering 702
  - classes 723
  - creating 696
  - deleting 701
  - external logins 703
  - listing properties 703
  - transaction management 713
- remote tables
  - about 695
  - listing 702
  - listing columns 707
- REMOTEPWD
  - using 812
- renaming database files 643
- replacing write server 666
- Replication Server
  - support 685
- reqtool
  - how to use 606
- requirements for using the debugger 818
- reserved words
  - remote servers 718
- RESIGNAL statement
  - about 431
- RESOURCE authority
  - about 552
  - granting 569
  - not inheritable 574
- resource planning
  - iqnumbercpus switch 52
- restore operations
  - about 639
  - displaying header file 651
  - ensuring correct order 647
  - excluding other users 650
  - performance issues 658
  - recovering from errors 651
  - SYSFILE after restore 641
  - to raw device 644
- RESTORE statement

- about 642
- restoring databases
  - renaming files 643
  - restoring the multiplex 661
  - restrictions
    - remote data access 718
- result sets
  - multiple 418
  - procedures 403, 417
  - variable 419
- ResultSet class 770
- ResultSetXml class 775
- RETURN statement
  - about 414
- REVOKE statement
  - about 573
- role of digital certificates
  - transport-layer security 593
- ROLLBACK statement 493
  - compound statements 410
  - procedures 434
- ROLLBACK TO SAVEPOINT statement 493
- routers
  - broadcasting over 161
- ROW DELIMITED BY option
  - LOAD TABLE statement 336
- row id
  - displaying 361
  - in notification message log 360
- ROW LOG
  - contents 349
  - example 350
  - LOAD TABLE option 341
  - LOG DELIMITED BY option 342
- rsaserver.crt 597
  
- S**
- SA\_DEBUG group
  - debugger 818
- sample database xxix, 11
- sample Java classes
  - JXml 761
  - OrderXml 761, 763
  - ResultSet 770
- ResultSetXml 761, 775
- sample.crt 597
- SAVEPOINT statement
  - and transactions 493
- savepoints
  - procedures 434
  - within transactions 493
- SAX
  - See also* XML parsers
  - Simple API for XML 762
- schedules 739–751
  - adding 749
  - definition components 742
  - recurrence frequency 742
- schemas
  - changing 239
  - creating 239
- scripts
  - IDebugAPI interface 830
  - IDebugWindow interface 834
  - writing debugger 829
- scrollable cursors
  - JDBC support 788
- scrolling
  - cursors 501
- SCSI tape backups 620
- section 508
  - compliance xxix
- secure socket layers 590
  - obtaining certificates for transport-layer security 606
- security 590
  - about 549
  - batch operations 563
  - FIPS 590
  - hiding objects 436
  - integrated logins 123, 124
  - login failures 559
  - procedures 397, 572, 581
  - Sybase IQ client architecture 592
  - views 581
- See database utilities 95
- SELECT DISTINCT projection 265
- SELECT permissions 570
- SELECT statement
  - delimiters 355

- in INSERT statement 356
- INTO clause 415
- JDBC 804
- join indexes 288
- objects 807
  - restrictions for view creation 251
- self-signed certificates
  - making for transport-layer security 598
  - transport-layer security 597, 599
- semicolon
  - command delimiter 435
- SendBufferSize communication parameter
  - description 170
- sensitivity
  - cursor behavior 501
- serialization
  - distributed computing 815
  - objects 814
- server
  - See* database server
- server address
  - DSEdit 682
- server authentication
  - transport-layer security 595
- server classes
  - 64-bit platforms 729
  - about 695
  - asajdbc 724
  - asaodbc 728
  - asejdbc 726
  - aseodbc 729
  - defining 695
  - ODBC 727
- server information
  - asasrv.ini file 116
- server log file 60
- server name
  - ODBC configuration 101
- server objects
  - adding 80
- server-authentication certificates
  - transport-layer security 606
- ServerName connection parameter 146
- ServerPort communication parameter 170
- servers
  - automatic startup on boot 32
- connecting 78
- converting query to write 666
- deleting 208, 211
- enabling for multiplex 194
- excluding 207
- including 207
- multiple databases on 690
- multiplex 41
- naming 47
- preserving configuration 208
- properties 208
- recycling 62
- restoring 661
- starting 207
- startup restrictions 35
- status 211
- stopping 207
- viewing properties 211
- Service Manager
  - starting servers 44
- SESSIONS communication parameter 171
- SET clause
  - UPDATE statement 392
- setAutocommit method
  - about 799
- setObject method
  - using 815
- seven-bit characters
  - about 507
- SGML
  - relation to HTML 753
  - relation to XML 753
- shell scripts
  - administrative 210
- shutdown
  - database 67
  - troubleshooting 127
- SIGNAL statement
  - procedures 426
- signed certificates
  - transport-layer security 604
- single-byte character sets
  - about 507
- single-node mode 243
  - example 667
- snapshot versioning

*See Also* versioning 471  
communication parameters  
    SendBufferSize 170  
SNDBUFSZ communication parameter  
    description 170  
software release number 23  
sort order  
    collations 503  
sort orders  
    definition 506  
sorting  
    collation file 536  
sp\_iqcheckdb  
    checking database consistency 637  
    DBCC\_LOG\_PROGRESS 638  
sp\_iqconnection  
    checking temporary space use 237  
sp\_iqdbspace  
    checking temporary dbspace use 237  
    dbspace usage information 229  
sp\_iqdbspaceinfo  
    dbspace usage information 229  
sp\_iquestdbspaces  
    estimating dbspace requirements 181  
sp\_iquestjoin 307  
    estimating join index space requirements 181  
sp\_iquestspace  
    estimating database space requirements 181  
sp\_iqindexinfo  
    displaying index information 230  
sp\_iprocedure  
    information about procedures 397  
sp\_iprocparm  
    procedure parameters 397  
sp\_iqrelocate  
    relocating objects 233  
specifying  
    drivers 78  
SQL Remote 476  
    login management 563  
    Message Agent 477  
    remote data access 718  
    starting 207  
    stopping 207  
sql.ini file  
    configuring 679

SQLCODE variable  
    introduction 424  
SQLCONNECT environment variable  
    connections 95  
SQLLOCALE environment variable  
    about 531  
    setting 541  
SQLSTATE variable  
    introduction 424  
Standardized General Markup Language  
    *See SGML*  
standards  
    section 508 compliance xxix  
standards and compatibility  
    section 508 compliance xxix  
star joins 302  
start line  
    ODBC configuration 101  
Start parameter  
    embedded databases 92  
START ROW ID option 359  
    about 359, 363  
    INSERT statement 354  
    partial-width inserts 356, 360  
start\_asiq utility 33  
    starting asiqdemo database 68  
starting 68  
starting databases  
    jConnect 812  
starting multiplexes 38, 41  
StartLine connection parameter 156  
startup  
    troubleshooting 127  
startup parameters 45  
startup script 33  
startup utility 33  
static collisions 212  
status  
    checking 200  
status icons 200  
stored procedures  
    about 10  
    Adaptive Server Enterprise 13  
    Adaptive Server Enterprise catalog 15  
    Adaptive Server Enterprise system 14  
    debugging 820

- displaying information about 397
- retrieving information 15
- retrieving parameter information 15
- Sybase IQ 11
- string and number defaults 454
- subtransactions
  - and savepoints 493
  - procedures 434
- SUM function 271
- Sybase Central 68
  - adding users to groups 576
  - altering tables 246
  - and permissions 571
  - column constraints 458
  - column defaults 451
  - creating dbspaces 218
  - creating groups 575
  - creating tables 239
  - creating users 567
  - creating views 251
  - dropping views 253
  - foreign keys 248
  - introduction 177
  - permissions 573
  - primary keys 247
  - starting 39, 41, 86, 194
  - stopping 70
  - system tables 249
- SYBASE environment variable
  - DSEDITION 681
- Sybase IQ
  - matching data types with Adaptive Server Enterprise 382
- Sybase IQ User Administration
  - list of procedures 558
  - overview 556
  - using with TDS connections 689
- Sybase IQ User Administration facility. *See Login Management*
- symbolic links 221
- synchronization
  - Sybase IQ server authentication 597
  - transport-layer security 589
  - transport-layer security with Sybase IQ 590
- synchronizing
  - about 295
  - synchronizing query servers 44
  - synchronizing the multiplex 204, 243
- SYS group 578
- SYSCOLAUTH view
  - permissions 587
- SYSCOLLATION table
  - collation files 535
- SYSCOLUMN table
  - integrity 469
- SYSDUMMY table
  - permissions 587
- SYSFILE table
  - file\_name after restore 641
- SYSFOREIGNKEY table
  - integrity 469
- SYSGROUP table
  - permissions 587
- SYSGROUPS view
  - permissions 587
- SYSINFO table
  - collation files 535
- SYSPROCAUTH view
  - permissions 587
- SYSPROCPERM table
  - permissions 587
- sysservers system table
  - remote servers 696
- sysservers table 696
- SYSTABAUTH view
  - permissions 587
- SYSTABLE table
  - integrity 469
  - view information 253
- SYSTABLEPERM table
  - permissions 587
- system events
  - trigger conditions 744
  - types 743
- system tables 16
  - about 249
  - character sets 535
  - indexes in 255
  - national languages 535
  - permissions 586
  - showing 214
- SYSCOLLATION 535

- SYSFILE 641
  - SYSINFO 535
    - users and groups 586
    - views 253
  - system views
    - integrity 469
    - permissions 586
  - system-level backups 634
  - SYSUSERAUTH view
    - permissions 587
  - SYSUSERLIST view
    - permissions 587
  - SYSUSERPERM table
    - permissions 587
  - SYSUSERPERMS view
    - permissions 587
  - SYSVIEWS view
    - view information 253
- T**
- table names
    - international aspects 515
    - local 706
    - procedures 435
  - table-level versioning
    - about 477
    - See Also* versioning 477
  - tables
    - adding keys to 247
    - altering 245
    - creating 239
    - defining proxy 706, 707
    - displaying columns 11
    - displaying size 13
    - dropping 246
    - group owners 577
    - join relationships 296
    - joining multiple 292
    - listing 15
    - listing remote 702
    - listing with sp\_iqtable 13
    - loading 324
    - locking 485
    - owner 552
- permissions 552
  - persistent 241
  - proxy 706
  - qualified names 577, 579
  - remote access 694
  - See Also* information in system tables 16
  - sysservers 696
- tabular data stream (TDS)
    - about 677
  - tape devices
    - for backup 625
- TCP/IP
    - addresses 682
    - connecting across firewalls 160
    - LDAP communication parameter 164
    - Open Server 686
    - server configuration 157
    - server port number 170
- TDS
    - about 677
  - TDS communication parameter 172
- TEMP environment variable
    - disk space 128
  - TEMP\_EXTRACT\_NULL\_AS\_EMPTY option 320
  - TEMP\_EXTRACT\_NULL\_AS\_ZERO option 320
- temporary storage
    - option to save space 501
- Temporary Store
    - about 7
- temporary tables
    - about 241
    - loading 242
    - versioning 484
- terminators
    - LOAD TABLE statement 327
- threads
    - use during loads 391
- TIME data type
    - optimizing loads 371
- time data types
    - matching Adaptive Server Enterprise and Adaptive Server IQ data 385
- TIME index 279
    - additional indexes 284
    - advantages 284
    - comparison to other indexes 284

- disadvantages 284
- recommended use 280
- TIMEOUT communication parameter 172
- times
  - procedures 436
- TMP environment variable
  - disk space 128
- TO communication parameter 172
- top table
  - size and performance 292
- TRACEBACK function 426
- transaction log
  - in system recovery 497
  - renaming 649
- transaction management 713
- transaction processing
  - about 471
- transactions
  - about 471
  - cursors in 500
  - definition 472
  - ending 473
  - in recovery 496
  - managing 714
  - procedures 434
  - remote data access 713
  - rolling back 495
  - savepoints 493
  - starting 473
  - subtransactions and savepoints 493
- Translation driver
  - ODBC configuration 99
- transport-layer security 589, 590
  - client architecture 592
  - invoking 596
  - obtaining certificates for transport-layer security 606
- trigger conditions
  - for system events 744
- triggering event handlers 750
- troubleshooting 208
  - database connections 110
  - debugging classes 823
  - remote data access 718
  - server address 684
  - server startup 129
- startup, shutdown, and connections 127
- Sybase IQ security 597
- TSQL\_HEX\_CONSTANT option
  - Open Client 687
- TSQL\_VARIABLES option
  - Open Client 687

## U

- UID connection parameter 156
- UNC connection parameter 156
- Unconditional connection parameter 156
- unenforced foreign keys 465
- Unicode character sets
  - about 530
- UNIQUE constraints 456
- UNIX
  - default character set 519
  - ODBC support 105
- unloading data
  - from pre-Version 12 Adaptive Server IQ 338
- UPDATE statement
  - using 391
  - using join operations 392
- upper code page
  - about 507
- URL
  - jConnect 811
- URL database
  - JDBC 811
- user accounts
  - adding with **sp\_addlogin** 14
- user administration
  - see Sybase IQ User Administration
- user IDs
  - creating 567
  - default 550
  - deleting 573
  - in message log 24
  - listing 586
  - managing 549
  - ODBC configuration 101
- user names. *See* user IDs
- USER special value
  - default 453

user-defined data types  
    CHECK conditions 458

user-defined datatypes  
    dropping 207

user-defined functions  
    calling 405  
    creating 404  
    dropping 406  
    execution permissions 406  
    parameters 413  
    using 404

Userid connection parameter 156

users  
    adding to groups 576  
    adding with **sp\_adduser** 14  
    creating in Sybase Central 567  
    creating individual 566  
    dropping 15  
    dropping with **sp\_droplogin** 14  
    locking 556  
    login failures 559

Using 105, 373

using a global certificate as a server certificate  
    transport-layer security 607

using a globally-signed certificate as an enterprise certificate  
    transport-layer security 608

using a self-signed certificate  
    transport-layer security 599

using chains of certificates  
    transport-layer security 594

using column defaults 447

using the signed certificates  
    transport-layer security 604

*util\_db.ini* file 564

utilities  
    Transaction Log 649

utility database  
    about 26  
    connecting 566  
    password to create databases 565  
    security 564  
    setting password 564  
    starting 564

V

validate option 208

VALUES option  
    INSERT statement 352

VARCHAR data  
    zero-length cells 374

variable width character sets  
    about 513

variable-length data transfer 620

variables  
    debugging 822

VERIFY communication parameter 173

verifying certificate fields  
    transport-layer security 608

verifying fields in certificate chains  
    transport-layer security 608

verifying passwords 568

version string 23

versioning  
    about 471, 477  
    at table level 477  
    cursors and 500  
    in system recovery 497  
    isolation levels 490  
    performance impact 498  
    temporary tables 484

vertical insertions  
    about 359

views  
    creating 250  
    deleting 252, 253  
    differences from permanent tables 250  
    inserting and deleting 251  
    modifying 252  
    owner 552  
    permissions 252, 552  
    security 581  
    SELECT statement restrictions 251  
    using 251  
    working with 249

Virtual Backup  
    decoupled 633  
    encapsulated 633

Visual Basic  
    ODBC configuration for 100

VLDB

managing 26

## **W**

WarehouseArchitect

about 176

warnings

procedures 429

WD index 277

advantages 279

disadvantages 279

recommended use 277

web servers

starting with transport-layer security 610

web services

starting with transport-layer security 610

WHERE clause

impact on index choice 266

join indexes 288

UPDATE statement 392

WHILE statement

syntax 408

white space characters

defined 538

WIN\_LATIN1 collation

about 527

Windows Service 32

wired memory

setting iqwmem switch 51

WITH GRANT OPTION clause 572

WORD SKIP option 332

working with column defaults in Sybase Central 451

write server

definition 9

failure 66

moving 666

replacing 666

comparison to HTML 753

customizable example 770

element storage 767, 778

in the database 753

installing in Adaptive Server IQ 754

introduction to 753

JXml sample class 761

OrderXml sample class 763

overview 754

parsers 762

relation to SGML 753

ResultSet sample class 770

ResultSetXml sample class 775

sample document 755

sample Java classes 761

source code for sample classes 754

specifying character set 756

use of CDATA section 782

XML data

element storage 760

XML documents

character sets 759

DTD 757

embedded DTD 757

formatting 759

formatting with XSL 755

invalid characters 782

parts of 756

valid 759

well-formed 757

XML format 311

XML operations

client side 761

server side 761

XML parsers 762

DOM interface 762

SAX interface 762

standard interfaces 762

-xs option

securing communications 610

XSL

Extensible Style Language 755

formatting XML 759

XML formatting instructions 755

## **X**

XML

accessing in SQL 761

additional information 754

CDATA section 782

## *Index*

### **Y**

year 2000  
conversion options 376

### **Z**

-Z option  
database server 129  
zeros  
converting to NULL 380