

SQLBase

Database Administrator's Guide

20-2121-1006



Trademarks

Centura, Centura Ranger, the Centura logo, Centura Web Developer, Gupta, the Gupta logo, Gupta Powered, the Gupta Powered logo, Fast Facts, Object Nationalizer, Quest, Quest/Web, QuickObjects, SQL/API, SQLBase, SQLConsole, SQLGateway, SQLHost, SQLNetwork, SQLRouter, SQLTalk, and Team Object Manager are trademarks of Gupta Technologies LLC and may be registered in the United States of America and/or other countries. SQLWindows is a registered trademark and TeamWindows, ReportWindows and EditWindows are trademarks exclusively used and licensed by Gupta Technologies LLC.

Microsoft, Windows and Visual Basic are either registered trademarks or trademarks of Microsoft Corporation in the United States of America and/or other countries.

Java is a trademark of Sun Microsystems Inc.

All other product or service names mentioned herein are trademarks or registered trademarks of their respective owners.

Copyright

Copyright © 2003 by Gupta Technologies LLC. All rights reserved.

SQLBase Database Administrator's Guide

20-2121-1006

August 2003

Contents

Preface.....	1-xvii
Audience.....	1-xviii
Organization of this manual	1-xviii
Syntax diagrams.....	1-xx
Notation conventions	1-xxi
Other helpful resources	1-xxii
Send comments to.....	1-xxii
1 RDBMS Concepts.....	1-1
Database administrator (DBA)	1-2
Organizing the DBA function	1-3
Relational databases.....	1-3
Representing relationships.....	1-3
Terminology	1-4
Relational operations	1-5
Comparison to other models.....	1-6
How SQL organizes data	1-6
Databases, tables, and columns.....	1-7
Indexes.....	1-8
Views	1-8
Synonyms.....	1-9
Stored commands and procedures.....	1-9
External functions.....	1-9
Triggers.....	1-10
System catalog tables.....	1-10
Concurrency and consistency.....	1-10

Databases and networks.	1-11
Distributed processing	1-11
The client-server model	1-11
Cooperative processing	1-12
Distributed database.	1-13
Benefits of distributed databases	1-15
2 Communication	2-1
Overview	2-2
Local connection.	2-3
Remote connection.	2-3
Supported protocols	2-3
Protocols supported by platform.	2-4
Client-to-server communication matrix.	2-4
Configuring SQL.INI	2-5
Connectivity Administrator	2-8
Configuring servers.	2-9
Configuring clients	2-11
Anonymous pipes	2-14
TCP/IP.	2-14
SPX.	2-15
SNMP	2-15
What is SNMP?	2-15
How SQLBase and SQLConsole use SNMP . . .	2-16
Database drivers and providers	2-19
COM+ and the SQLBase Resource Manager.	2-19
Configuring servers and clients	2-20
Shutting down SQLBase Resource Manager. . .	2-20
3 Configuration: SQL.INI and the Registry	3-1
SQLBase configuration file	3-2
Finding the configuration file.	3-2
Editing SQL.INI.	3-3

SQL.INI format	3-3
Communications	3-3
dbdfault section.	3-4
Configuration entries	3-4
Command line keywords	3-4
Syntax rules	3-4
When do keyword changes take effect?	3-6
Keyword list.	3-6
Keyword descriptions	3-10
ansijoin syntax	3-11
audit	3-11
autostartserverpath.	3-11
batchpriority	3-11
cache	3-11
centurydefaultmode	3-12
character set	3-13
clientcheck	3-14
clientname	3-14
clientruntime dir	3-15
cmdtimeout.	3-15
comdll	3-15
commitserver	3-17
connectpauseticks	3-17
connecttimeout	3-17
country	3-18
dbdir	3-19
dbname.	3-20
dbwin	3-22
defaultdatabase	3-23
defaultpassword	3-23
defaultuser	3-23
defaultwrite	3-23

directsap	3-25
disablelogspacecheck.	3-25
displevel	3-26
errorfile	3-26
extdll	3-27
fileaccess	3-28
groupcommit.	3-29
groupcommitdelay	3-29
inmessage	3-30
insertioncontext.	3-31
listenport	3-31
listenretry	3-32
locks	3-32
locktimeout	3-32
log	3-32
logdir	3-33
logfileprealloc	3-34
mainwin.	3-34
maxnestinglevel	3-35
ndsloginid	3-35
ndsloginpassword.	3-36
negotiateapi	3-36
netcheck	3-36
netchecktype.	3-37
netlog	3-37
nwadvertisemode	3-38
optimizefirstfetch.	3-38
optimizerlevel	3-39
oracleouterjoin	3-39
osavgwindow	3-40
ossamplerate	3-41
outmessage	3-41

partitions	3-42
password	3-42
preferrednameservice	3-42
procwin	3-43
readonly	3-44
searchcontext	3-44
secureapi	3-46
servername	3-46
servernames	3-47
serverpath	3-47
showmaindbname	3-49
silentmode	3-49
sortcache	3-49
statwin	3-50
syswin	3-50
tempdir	3-50
threadmode	3-51
threadstacksize	3-52
timeout	3-52
timestamps	3-52
timezone	3-53
users	3-53
workalloc	3-53
worklimit	3-54
The Registry	3-54
4 Databases	4-1
About databases	4-2
Database template (start.dbs)	4-2
Database names	4-2
Database size	4-3
Database location	4-4
Database cache	4-5

Pages	4-6
Committing	4-6
Checkpoint time interval	4-6
Database files	4-7
Transaction log files	4-7
Temporary files	4-13
Partitions	4-15
What is a partition?	4-15
What are database areas?	4-16
What are storage groups?	4-17
The MAIN database	4-17
Database auditing	4-18
Audit file	4-19
Starting and stopping an audit	4-20
Types of audit operations	4-20
Record formats	4-22
Sending a message to an audit file.	4-25
Audit file examples	4-26
5 DBA Interfaces	5-1
Client programs.	5-2
SQLTalk	5-2
Application Programming Interface (API)	5-3
SQLConsole	5-3
Character-based server interface	5-4
Server Status display screen (F1)	5-4
Process Activity display screen (F2).	5-6
System Activity display screen (F3)	5-8
NetWare specifics.	5-12
SQLBase Server Console	5-12
Server Status	5-17
Databases.	5-17
Process Activity	5-17

System Activity	5-18
Configuration	5-21
SQLBase Management Console (SMC)	5-21
SMC information displays	5-22
SMC actions are performed by context menus.	5-23
.	5-26
6 DBA Operations	6-1
Starting and stopping SQLBase	6-2
NetWare	6-2
Windows	6-3
Setting configuration keywords.	6-4
Creating a database	6-5
Deleting a database	6-6
Deinstalling a database.	6-6
Re-installing a database	6-8
Loading and unloading a database	6-8
Loading	6-9
Unloading	6-9
Segmenting a LOAD/UNLOAD file	6-10
Error logging and recovery	6-14
Load performance enhancements	6-14
Creating a read-only database	6-16
Partitioning a database	6-17
Step 1: creating a database area	6-18
Step 2: creating a storage group	6-18
Step 3: creating a database	6-18
Maintaining a partitioned database	6-19
Database areas	6-19
Storage groups	6-20
Extents	6-20
Free space in partitioned databases.	6-21
Reorganizing a database	6-21

Checking database integrity	6-22
Running multiple versions of SQLBase.	6-23
Version 8.5	6-23
Versions prior to 8.5	6-24
7 Security and Authorization	7-1
Database authority	7-2
Creating a user account	7-3
Changing a user's password	7-4
Revoking a user's authority	7-5
Table and view privileges	7-6
Granting privileges	7-6
Revoking privileges.	7-8
Synonyms	7-9
Creating a synonym	7-9
Dropping a synonym.	7-11
Views.	7-11
Creating a view.	7-12
Dropping a view	7-14
Execute privileges for stored procedures	7-14
External function privileges.	7-14
System catalog	7-15
Granting access to the system catalog.	7-15
Server security	7-16
Server connection password	7-16
Server security password	7-17
Database page encryption	7-20
None	7-21
Low	7-21
Medium	7-22
High	7-22
Transaction log encryption	7-22
REORGANIZE with encrypted databases	7-22

Database page alteration protection	7-22
ALTER DBSECURITY	7-24
ALTER EXPORTKEY	7-24
Backups and security configuration settings.	7-25
Transmission security	7-25
secureapi keyword	7-26
negotiateapi keyword	7-27
Delayed password validation	7-27
Remote file protection	7-27
Tutorial	7-28

8 Backing Up and Restoring Databases	8-1
Making a backup plan	8-2
Crash recovery	8-2
Media recovery	8-2
Backing up a database	8-3
Online backups.	8-3
Offline backups.	8-6
SQLTalk commands	8-7
Restoring a database	8-8
Restoring an online backup	8-8
Restoring an offline backup	8-11
Examples	8-12
Example 1 - Snapshot	8-12
Example 2 - Backing up and restoring database and log files separately.	8-12
Example 3 - Performing Incremental Backups with Backup Snapshot	8-14
Backing up and restoring partitioned databases.	8-16
Online backup.	8-16
Offline backup.	8-16
The MAIN database	8-17
Backups with secure databases	8-17

9 Distributed Transactions	9-1
What is a distributed transaction?	9-2
Setting up a distributed transaction.	9-3
Using SQLTalk	9-4
Using the SQL/API	9-5
Other issues	9-5
Two-phase commit	9-6
Components	9-7
Process	9-8
Failure recovery	9-13
In-doubt transactions	9-13
Manually resolving transactions	9-13
Performance issues	9-15
10 National Language Support	10-1
NLS configuration files	10-2
SQL.INI	10-2
country.sql	10-3
country.dbs and country.tlk.	10-4
error.sql	10-4
message.sql	10-4
country.sql directives.	10-5
#alpha	10-5
#firstbyte	10-5
#identifier	10-5
#lower	10-6
#numeric	10-6
#prefix	10-7
#punctuation	10-7
#secondbyte	10-7
#translate	10-8
#upper	10-9
#whitespace	10-9

Building a database with NLS	10-9
11 SQLBase and NetWare Directory Services(NDS)	11-1
SQLBase and NDS	11-2
Configuring SQLBase Server for NetWare for NDS	11-2
Insertion context	11-2
NDS logon ID and password	11-5
Advertising mode	11-5
SQLBase NDS schema extension	11-6
SQLBase classes	11-7
SQLBASE::DBServer	11-8
SQLBASE::Database	11-8
Setting bindery emulation	11-9
Configuring Clients for NDS	11-10
NDS logon ID	11-10
Search order	11-10
Search context	11-11
12 Running SQLBase Server as a Windows Service	12-1
About service programs	12-2
Installing SQLBase Server for Windows	12-2
Administering service programs	12-3
Using the Services window of the Control Panel	12-3
Service information in the Windows Registry	12-4
Windows Application Event Log	12-5
Shutting down SQLBase Server for Windows	12-8
Running SQLBase as an application program	12-8
A System Catalog Tables	A-1
System catalog summary	A-2
SYSADM.SYSCOLAUTH	A-3
SYSADM.SYSCOLUMNS	A-3

SYSADM.SYSCOMMANDS	A-5
SYSADM.SYSDEPENDENCIES	A-6
SYSADM.SYSEVENTS	A-6
SYSADM.SYSEXECUTEAUTH	A-7
SYSADM.SYSEXTFUN	A-7
SYSADM.SYSEXTPARAM.	A-8
SYSADM.SYSFKCONSTRAINTS	A-8
SYSADM.SYSINDEXES.	A-9
SYSADM.SYSKEYS.	A-11
SYSADM.SYSOBAUTH	A-11
SYSADM.SYSOBSYN	A-12
SYSADM.SYSPARTTRANS.	A-13
SYSADM.SYSPKCONSTRAINTS	A-14
SYSADM.SYSROWIDLISTS	A-14
SYSADM.SYSSYNONYMS	A-14
SYSADM.SYSTABAUTH	A-15
SYSADM.SYSTABCONSTRAINTS	A-15
SYSADM.SYSTABLES.	A-16
SYSADM.SYSTRGCOLS.	A-18
SYSADM.SYSTRIGGERS	A-19
SYSADM.SYSUSERAUTH.	A-20
SYSADM.SYSVIEWS.	A-20
System table operations	A-20
System catalog views	A-21
Server-independent system catalog views	A-22
MAIN database system tables	A-24
SYSADM.CSVPARTS	A-25
SYSADM.CSVTRANS	A-25
SYSADM.DEFAULTS.	A-26
SYSADM.DATABASES	A-26
SYSADM.AREAS	A-27
SYSADM.STOGROUPS.	A-27

SYSADM.STOAREAS	A-27
SYSADM.EXTENTS	A-27
SYSADM.FREEEXTS	A-28
MAIN database system views	A-28
MAIN stored commands	A-29
B SQLBase Procedures	B-1
SQLBase-supplied procedures	B-2
RECOMPILE procedure	B-2
SYSADM.SYSPROC_ALTTABTRIG procedure ..	B-3
C System Utility Tables	C-1
System utilities table summary	C-2
SYSCOMMITORDER	C-2
Glossary	Glossary-1
Index	Index-1

Preface

This manual provides reference information about setting up and maintaining SQLBase databases.

This preface provides the following information:

- Who should read this manual.
- The organization of this manual.
- The documentation format.
- The notation conventions used in this manual.
- Related publications.

Audience

This manual is intended for:

- **Application Developers** building client applications that access databases using Gupta frontend products like SQLTalk, Team Developer, and the SQL/API.
- **Database Administrators** performing day-to-day operation and maintenance of the database and configuring the database settings to customize the environment.
- **System Administrators** maintaining and configuring the operating system, such as configuring the physical and logical devices on NetWare used to install SQLBase. This includes maintaining operating system access security.
- **Network Administrators** maintaining and configuring the network. This includes network access security, logical and physical network address assignment, and network resource allocations.

This manual assumes you have a working knowledge of relational databases and SQL.

Organization of this manual

This manual is organized into the chapters listed below.

Chapter 1: RDBMS Concepts

This chapter provides an overview of a DBA's responsibilities and relational database management systems.

Chapter 2: Communications

This chapter summarizes your communication options and defines each platform's client and server executables.

Chapter 3: Configuration: SQL.INI and the Registry

This chapter provides details for each configuration file (SQL.INI) keyword and explains how SQLBase uses the Registry.

Chapter 4: Databases

This chapter discusses database issues such as cache, temporary files and log files, partitioning, and auditing.

Chapter 5: DBA Interfaces

This chapter introduces the SQLBase client software components that you use to perform DBA functions, such as SQLTalk, the SQL/API, SQLBase Server Console, and SQLBase Management Console (SMC).

Chapter 6: DBA Operations

This chapter touches on all aspects of a DBA's daily operations.

Chapter 7: Security and Authorization

This chapter shows you how to implement security and authorization through database authority, table and view privileges, and server security. This chapter also discusses advanced security features such as database page encryption, database page alteration protection, transmission security, and delayed password validation.

Chapter 8: Backing Up and Restoring Databases

This chapter discusses SQLBase's options for backing up and restoring databases.

Chapter 9: Distributed Transactions

This chapter describes how to create distributed transactions.

Chapter 10: ODBC Support

This chapter describes SQLBase's support for the Microsoft Open DataBase Connectivity (ODBC) standard.

Chapter 11: National Language Support

This chapter describes how you customize SQLBase for non-English languages.

Chapter 12: NetWare Directory Service Support

This chapter describes SQLBase's support for NetWare Directory Service (NDS).

Chapter 13: Running SQLBase Server as a Windows Service

This chapter describes running SQLBase Server as an application program or a Windows service program.

Appendix A

This appendix describes the contents of each table in the SQLBase system catalog.

Appendix B

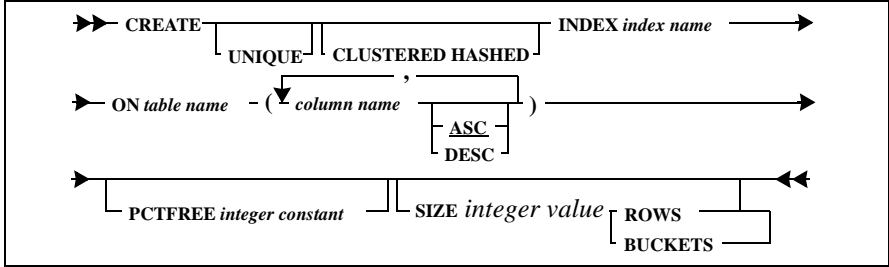
This appendix describes SQLBase system procedures.

Appendix C

This appendix describes the SQLBase system utility tables.

Syntax diagrams

This manual uses syntax diagrams to show how to enter commands. The syntax for the CREATE INDEX command is used here as an example.



Read the syntax diagram from left to right and top to bottom.

The line with the command name (CREATE) is the main line of the command. Mandatory keywords and arguments (such as INDEX or ON *table name*) appear on the main line or a continuation of the main line.

This example diagram could generate the commands shown in these examples:

```

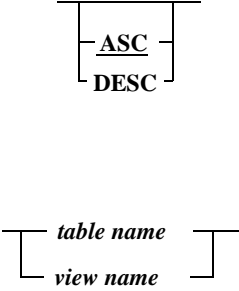
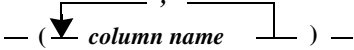
CREATE UNIQUE INDEX PARTIDX ON PARTS (PARTNO);

CREATE INDEX ORDER$DATE$IDX ON ORDERS (ORDERNUM, DATE);

```

The following table shows the syntax diagram symbols used in this manual.

Symbol	Description
>>>	A double arrow pointing right means the start of a command.
→	A single arrow pointing right means a continuation line of a command.
<<<	The double arrow pointing left means the end of a command.
[UNIQUE]	Optional clauses and keywords (such as UNIQUE) hang off the main or continuation lines.

Symbol	Description
	<p>If there is an optional item with alternate choices, the choices are in a vertical list. In this example, ASC and DESC are alternate non-mandatory options. ASC is underlined, which means it is the default and can be omitted.</p> <p>If an item is mandatory, the first alternative is on the main line (this example is from the UPDATE command).</p>
	<p>When you can repeat arguments of the same type (such as a list of column names), an arrow pointing downward is suspended above the argument. A delimiter or operator on this line shows what separates each argument (such as commas separating column names).</p>

Notation conventions

Before you start using this manual, it is important to understand the typographical conventions we use in this manual:

Formatting Convention	Type of Information
You	A developer who reads this manual
User	The end-user of applications that you write
bold type	Menu items, push buttons, and field names. Things that you select. Keyboard keys that you press.
Courier 9	Builder or C language code example
SQL.INI MAPDLL.EXE	Program names and file names
Precaution	Warning:
Vital information	Important:
Supplemental information	Note:

Formatting Convention	Type of Information
Alt+1	A plus sign between key names means to press and hold down the first key while you press the second key

Other helpful resources



Gupta Books Online. The Gupta document suite is available online. This document collection lets you perform full-text indexed searches across the entire document suite, navigate the table of contents using the expandable/collapsible browser, or print any chapter. Open the collection by selecting the Gupta Books Online icon from the **Start** menu or by double-clicking on the launcher icon in the program group.

World Wide Web. Gupta Software's world wide Web site contains information about Gupta Technologies LLC's partners, products, sales, support, training, and users. The URL is <http://www.guptaworldwide.com>.

The technical services section of our Web site is a valuable resource for customers with technical support issues, and addresses a variety of topics and services, including technical support case status, commonly asked questions, access to Gupta's online newsgroups, links to shareware tools, product bulletins, white papers, and downloadable product updates.

Our Web site also includes information on training, including course descriptions, class schedules, and certified training partners.

Send comments to...

Anyone reading this manual can contribute to it. If you have any comments or suggestions, please send them to:

Technical Publications Department
Gupta Technologies LLC
975 Island Drive
Redwood Shores, CA 94065

or send email, with comments or suggestions to:

techpubs@guptaworldwide.com

Chapter 1

RDBMS Concepts

This chapter briefly explains some concepts that you need to understand in order to use *SQLBase* software:

- A DBA's responsibilities.
This section discusses the responsibilities of a Database Administrator.
- Databases and networks
This section describes client/server terminology as it applies to *SQLBase*.

Database administrator (DBA)

As a DBA, you ensure the smooth operation of one or more databases and you are responsible for the design, planning, installation, configuration, security, management, maintenance, and operation of a database management system (DBMS) and its supporting network.

A good way to ensure that the databases meet the needs of your organization is by becoming familiar with the organization's applications: who are its users, what data is stored and accessed, and what types of transactions occur.

A DBA's specific responsibilities include:

- Keeping the database servers running on a daily basis.
- Diagnosing and resolving system problems.
- Installing both database server and client software.
- Creating databases.
- Backing up and recovering databases.
- Controlling security and access to the database and its objects.
- Monitoring and tuning the performance of a database as well as the applications that access it.
- Managing communications.
- Managing disk space and estimating storage needs.
- Advising developers about table, index, and view design, multi-user considerations, networking, and loading, converting, and unloading data.
- Ensuring data availability, accuracy, completeness, integrity, and consistency.
- Auditing the use of a database.
- Administering the system catalog.
- Helping database application users.

Organizing the DBA function

There are two SQLBase authority levels that a DBA can have:

- **SYSADM**

The most powerful authority level. There should only be one user with SYSADM authority. This user can designate one or more other users as DBAs.

- **DBA**

A user with this authority level has privileges on all database objects and can grant, change, or revoke the object privileges of any user but SYSADM.

Access to the SYSADM and DBA accounts should be tightly controlled.

On an organizational level, your company needs to determine the number of people required to maintain its databases and applications. Depending on the size of the databases and the applications, you may have one or more DBAs. They, in turn, may specialize in maintaining databases *or* applications, or they may support *both* a database and the applications that access it.

If you work in a large organization, you are likely to have one user with SYSADM authority who designates several users with DBA authority level and privileges. The DBAs would then share the responsibility for the database maintenance tasks.

If you work in a small organization, the user with SYSADM authority may be the DBA as well.

Read *Chapter 7, Security and Authorization* for more information on authority levels.

Relational databases

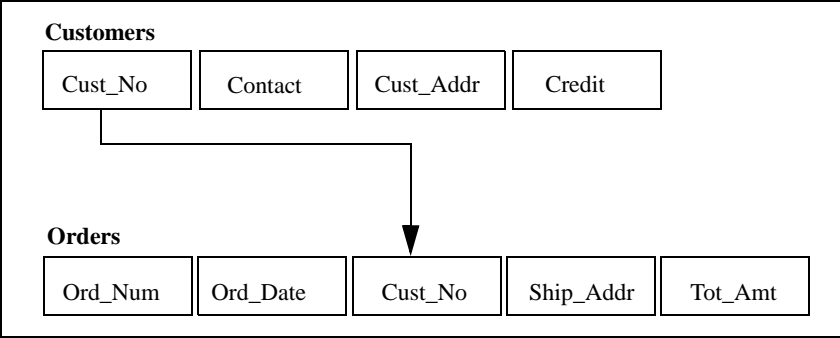
Users need data that is easy to access and that they can use in both ad-hoc queries and daily transaction processing. The data structures should be flexible.

A relational database reduces data to its most basic level in simple, two-dimensional tables that contain columns and rows of data values. Tables are easy to visualize. Users can explain their needs in easy-to-understand terms. Data modeling is flexible.

Representing relationships

A collection of tables can represent complex data relationships. The relationship between two tables is defined by a column that both tables contain. This makes the links between the tables easy to see and understand.

For example, in an order entry system, the customer information is stored in a separate table than the order information. The two tables are related by the existence of a common column: customer number (Cust_No).



Customers and Orders tables in an order entry system

This link enables SQLBase to match a row in the Orders table to its associated row in the Customers table, allowing you to find the customer name and address for each order.

Terminology

The terms used to refer to database objects vary from system to system. The following table shows how the terms relate to each other. Gupta uses the “Relational database” terms.

Relational database	Set theory	Conventional data processing
Table	Relation	File or data set
Row	Tuple	Record
Column	Attribute Domain	Field

The word *relation* comes from the study of data structures done by E.F. Codd and others at IBM. Codd used set theory. A set is a group of objects with a defining rule that enables you to tell whether a given object is in the set or not. For example, the set of white rabbits is a subset of all rabbits with the defining property that the animal has white fur.

A relational system produces a result set by retrieving a set of rows from one or more tables. You can, in turn, operate on this result set instead of operating against the

tables from which you originally retrieved the rows. This set-at-a-time approach means that a single relational command can retrieve, update, or delete multiple rows stored in the database.

The sequence (left-to-right or up-to-down) of data in a table holds no information about the data itself. Part of the flexibility of a relational database comes from this property; no information is “hidden” in the physical layout of the database.

In a relational system, the meaning of a column or row does not depend on its order relative to the other columns and rows. This means that you can retrieve data from a select number of columns without having to retrieve data from all columns. The same goes for rows: you can retrieve data from a subset of rows without having to retrieve all the rows.

Relational databases reduce the storage of redundant data. This has three benefits:

- You can rearrange data and combine it easily into new relationships; it is not locked into certain relationships because of the way it is stored.
- You can more easily update data because there are fewer instances of the data. This reduces the likelihood of errors arising from a failure to update all instances.
- It reduces disk space requirements.

Relational operations

Codd's relational algebra defined three key data retrieval operations.

Selection

Selection is the process of retrieving all rows of a table which meet some specified criterion. Note that this is a very narrow definition of selection, and should not be confused with the SQL `SELECT` command, which does more than just selection.

Projection

Projection is the process of retrieving one or more columns from a table in a specified order and removing any duplicate rows.

Join

Joining is the process of retrieving columns from different tables where the values of a common column in each table are equal.

The ability to join two or more tables is powerful. The join operation distinguishes relational systems from non-relational systems.

The SQL `SELECT` command performs selection, projection, and join operations.

Comparison to other models

Hierarchical and network database models are procedural and record-at-a-time. To find a record, you have to navigate or find a path to the record you want and give multiple procedural commands that tell the system to walk down that path, step-by-step. You need a detailed understanding of how the data is stored. Once you have created a database and loaded data into it, it can be difficult to change.

In contrast, a relational system provides automatic navigation. You do not have to know how the database stores data in order to retrieve, change, or destroy that data. This makes it easy to access data.

Hierarchical and network database systems store some data as values and other data as pointers. For example, in a network system, the data that order #123 comes to a total of \$58.00 would be stored as a value in a field. The data that order #123 belongs to customer #345 would be stored as a pointer from the order record to the corresponding customer record.

You must use pointers to associate records to one another in a network or hierarchical database. You must decide whether to store data as a pointer or as a value when you define the database. Network and hierarchical systems use relationships that are pointer-based and predefined or static.

In contrast, relational database systems can use any value to link one table to another, and you define relationships between values when you query the data, not when you create the tables. This gives you maximum flexibility to create spontaneous queries.

How SQL organizes data

SQL objects include:

- Databases
- Tables
- Columns
- Indexes
- Views
- Synonyms
- Stored commands
- Stored procedures
- External functions
- Triggers

Databases, tables, and columns

A *database* is a set of SQL objects. When you create a database, you are actually naming an eventual collection of tables, associated indexes, and views.

A single database can contain all the data associated with one application or with a group of related applications. Collecting data into a single database lets you enable or prevent access to all the data in just one operation.

A database contains one or more tables. Each table has a name and contains a specific number of *columns* and *rows*. Each column in a row is related in some way to the other columns in the same row.

Column		
Cust_No	Contact	Credit
46372986	E. Smith	\$3000.00
12162344	R. Vince	\$1500.00
98121735	G. Handle	\$ 580.00
55421888	B. Harty	\$2000.00
89923942	S. Jones	\$ 550.00

Each column has a name and a data type. Data values exist at the intersection of a row and a column.

Data uniqueness

In theory, no row should be a duplicate of any other row in the same table. Consider a sales order table with columns for the date, product code, quantity, and customer ID. If a customer orders ten widgets in the morning and then another ten in the afternoon, this would create two duplicate rows in the table, one for each order.

To ensure the creation of unique rows, you might create an additional column to store the time at which the order was placed, or to store a uniquely-generated sequence number (such as an invoice number).

A table can have a primary key which is a column or a group of columns whose value uniquely identifies each row. Another table can have a foreign key which is a column or a group of columns whose values are equal to those of the first table's primary key. One of the rules of referential integrity is that the value of a table's foreign key must match the value of another table's primary key.

Changing the database structure

SQLBase has SQL commands that enable you to add new columns to existing tables and make existing columns wider. These changes take effect immediately and no database reorganization is necessary.

Indexes

An *index* is an ordered set of pointers to the data in a table. It is based on the data values in one or more columns of the table. SQLBase stores indexes separately from tables.

SQLBase decides whether or not to use an index when accessing a table, and users need not be aware that SQLBase is using an index.

An index provides two benefits:

- It improves performance because it makes data access faster.
- It ensures uniqueness. A table with a unique index cannot have two rows with the same values in the column or columns that form the index key.

Views

A *view* is an alternate way of representing data that exists in one or more tables. A view can include all or some of the columns from one or more *base tables*. You can also base a view on other views or on a combination of views and tables.

A view looks like a table and you can use it as though it were a table. You can use a view name in a SQL command as though it were a table name. You cannot do some operations through a view, but you do not need to know that an apparent table is actually a view.

A table has a storage representation, but a view does not. When you store a view, SQLBase stores the definition of the view in the system catalog, but SQLBase does not store any data for the view itself because the data already exists in the base table or tables.

A view lets different users see the same data in different ways. This allows programmers, database administrators, and end users to see the data as it suits their needs. For example, employees might have access to columns containing general information about office locations and phone extensions, while administrators might have access to additional columns containing more confidential information such as employees' home addresses and phone numbers.

Synonyms

A *synonym* is another name for a table, view, or external function. When you access a table, view, or external function created by another user (once you have been granted the privilege), you must fully-qualify the table name by prefixing it with the owner's name, unless a synonym for the table or view is available. If one is available, you can refer to the user's table or view without having to fully qualify the name.

Stored commands and procedures

A *stored command* is a compiled query, data manipulation command, or procedure that is stored for later execution. SQLBase stores the command's or procedure's execution plan as well, so subsequent execution is very fast.

A SQLBase procedure is a set of Scalable Application Language (SAL) and SQL statements that is assigned a name, compiled, and optionally stored in a SQLBase database. Procedures reduce network traffic and simplify your applications since they are stored and processed on the server. They also provide more flexible security, allowing end users access to data which they otherwise have no privilege to access.

SQLBase procedures can be static or dynamic. *Static procedures* must be stored (at which time they are parsed and precompiled) before they are executed. *Dynamic procedures* contain dynamic embedded SQL statements, which are parsed and compiled at execution time.

SQLBase also provides preconstructed procedures as useful tools to help you maintain your database. See *Appendix B* for a description of SQLBase-supplied procedures.

External functions

An *external function* is a user-defined function that resides in an “external” DLL (Dynamic Link Library) that is invoked within a SQLBase stored procedure. SQLBase accepts external functions in the language of your choice, such as C and C++. The SQLBase server converts data types of parameters that are declared in stored procedures into their external representation.

Using external functions enhances the power of the SQLBase server, allowing you to achieve maximum flexibility and performance with minimal programming effort. It extends the functionality of stored procedures with no impact on the application or the server. When external functions are called, they are dynamically plugged in and behave like built-in functions. For details, read *Chapter 8, External Functions* in the *SQLBase SQL Language Reference*.

Triggers

A *trigger* activates a stored or inline procedure that SQLBase automatically executes when a user attempts to change the data in a table. You create one or more triggers on a table, with each trigger defined to activate on a specific command (an INSERT, UPDATE, or DELETE). You can also define triggers on stored procedures.

Triggers allow actions to occur based on the value of a row before or after modification. Triggers can prevent users from making incorrect or inconsistent data changes that can jeopardize database integrity. They can also be used to implement referential integrity constraints. For details on referential integrity, read *Chapter 6, Referential Integrity* in the *SQLBase SQL Language Reference*.

For details on the trigger execution order before a single data manipulation statement is executed, read the Section *DML Execution Model* in *Chapter 1* of the *SQLBase SQL Language Reference*.

System catalog tables

SQLBase maintains a system catalog, or *data dictionary*, in every database. The system catalog contains tables that store information about the database's tables, views, columns, indexes, execution plans, and security privileges.

When you create, change, or drop a database object, SQLBase modifies rows in the system catalog tables that describe the object and how it is related to other objects.

The system catalog also contains the name, size, type, and valid values of each table's columns.

You can query the system catalog tables just like any other table.

Read *Appendix A, System Catalog Tables* for detailed information about the data dictionary tables.

Concurrency and consistency

To guarantee the integrity of data, SQLBase uses page locks to prevent users from changing data while another user is reading or changing data on the same page. Locks prevent lost updates from occurring and ensure that data does not change while you are reading it or changing it yourself.

SQLBase stores data in 1K pages, and can lock both base table pages and index pages. For details, read the section *Database locking* in *Chapter 4, About Databases*.

Databases and networks

This section describes client/server terminology as it applies to SQLBase.

Distributed processing

The term *distributed processing* describes the placement of computers where they are needed, whether on the floor of a manufacturing facility, in an accounting department, in a laboratory, at a field location, or in an executive office. A network links computers so that they can exchange information, share resources, and perform distributed processing.

A distributed processing network allows information to originate anywhere in the network. You can place systems where they are needed while still having access to the facilities of other widely dispersed systems. Information can be exchanged among all parts of an organization.

The term distributed processing had been used for several years to label network configurations where remote computers exchanged data. Two developments in the 1980s brought a new meaning to the term: PCs (Personal Computers) and LANs (Local Area Networks).

PCs supplied solutions to many simple information processing requirements. The user interface on PCs became sophisticated with bit-mapped displays that used windowing and graphics, and a mouse. This made applications easy to use.

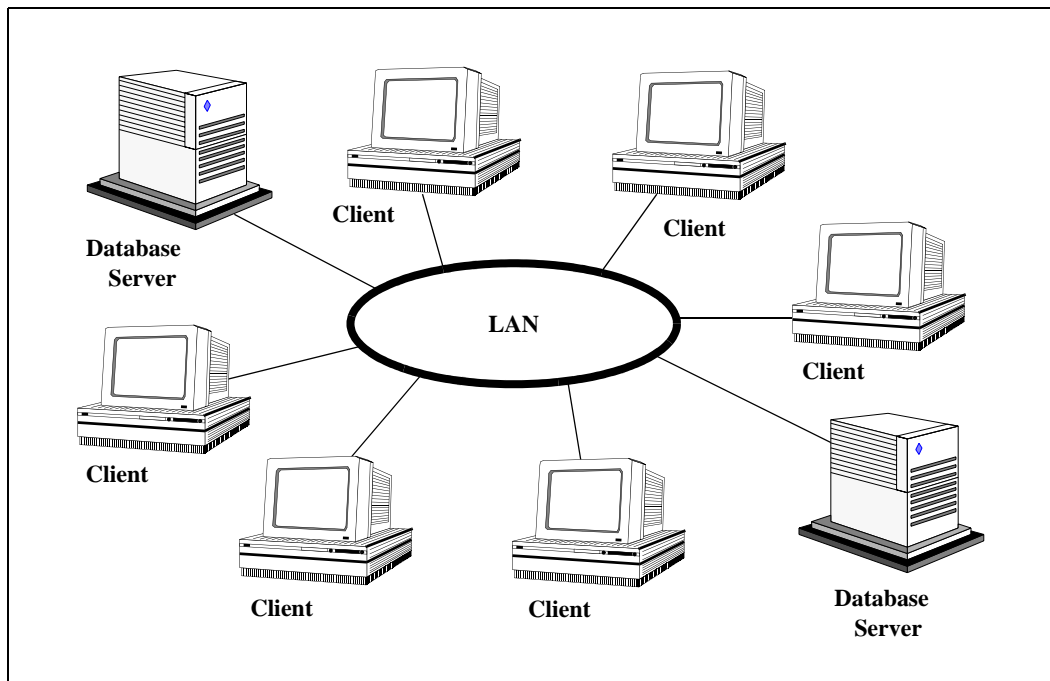
With a LAN, PCs could communicate with each other and share resources such as databases. Users began to wonder if they could move some transaction processing systems, which had traditionally run on minicomputers and mainframes, onto PCs. They also began to think about new types of applications that had not been possible before.

The client-server model

Client-server technology is a form of distributed processing where computing activities are shared among cooperating, networked computers. An application is functionally split into two or more programs which execute on different computers and communicate with each other by passing messages across the network.

The client programs run on users' PCs. Server programs run on more powerful computers. The client sends a request to a server and when the server receives the request, it processes it and sends the results back to the client.

Typically, a local-area network contains more clients than servers. Client computers cannot share their resources or use the resources of other client computers.



Through client-server architecture, users gain access to capabilities available on a server computer which are not available locally on their desktop computer. A client-server application has the capabilities of both the client and the server.

Servers are usually the most powerful computers on the local area network. Certain criteria make some systems better suited to be servers, such as a high input/output transfer rate and network throughput (the speed at which a server processes network tasks). High disk capacity and multiple disk controllers are another criteria. A server is usually multi-tasking so that it can serve multiple clients simultaneously.

Cooperative processing

When the processing duties are shared by a client and server in a sophisticated manner, it is called *cooperative processing*. An example of cooperative processing is a distributed database system. A client takes user input and submits processing requests to the server. Database processing, such as retrieving and sorting data, is centralized at a database server. Application processing is distributed between the clients and the server.

The client and server cooperate to do a job. The client manages the screen display and graphics, carries on an interactive dialogue with the user, validates data, does some local computation, and provides a network interface that talks to the server. The

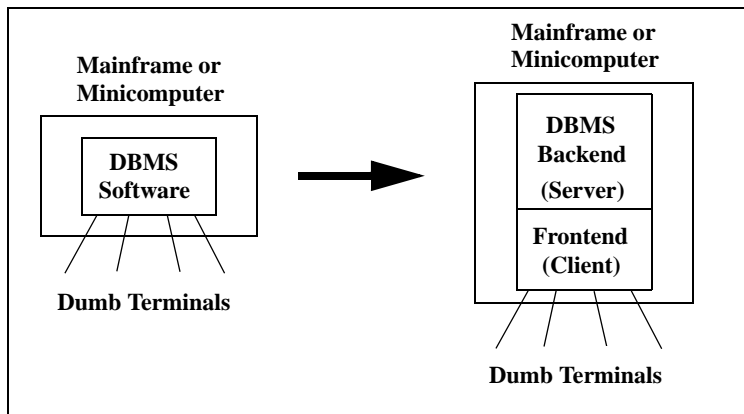
database server is responsible for data storage, security and integrity, and input/output.

In a cooperative processing environment, components of an application run where they are best suited. Data-intensive applications suffer performance problems when all computing happens on the client. The idea is to off-load time- and data-intensive tasks to the database server while doing as much processing on the client as possible. It is often the case that requests are processed faster and more economically on a shared high-performance database server than on a stand-alone client computer.

By their nature, some applications are suited to cooperative processing. For example, a *network-intrinsic* application is one that does not make sense on a stand-alone system. Communications is an integral part of the overall process of completing the application. Distributed database is a good example of a network-intrinsic application because it uses the network to provide *shared* access to data for many users.

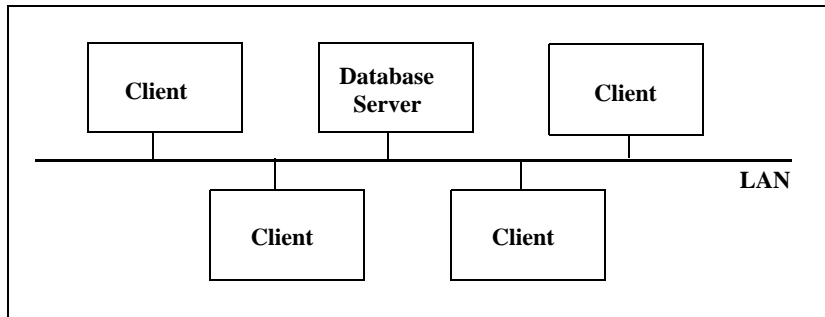
Distributed database

Database software has evolved. At first, database software was monolithic and ran on one large computer accessed by dumb terminals. Later, database software became more sophisticated and split the database function into two components: the client (frontend) and the server (backend). Note that both components still ran on the same mainframe or minicomputer at this time.



The start of database software evolution

Eventually, database software vendors moved the client component down to microcomputers and LANs, and the database server soon followed.



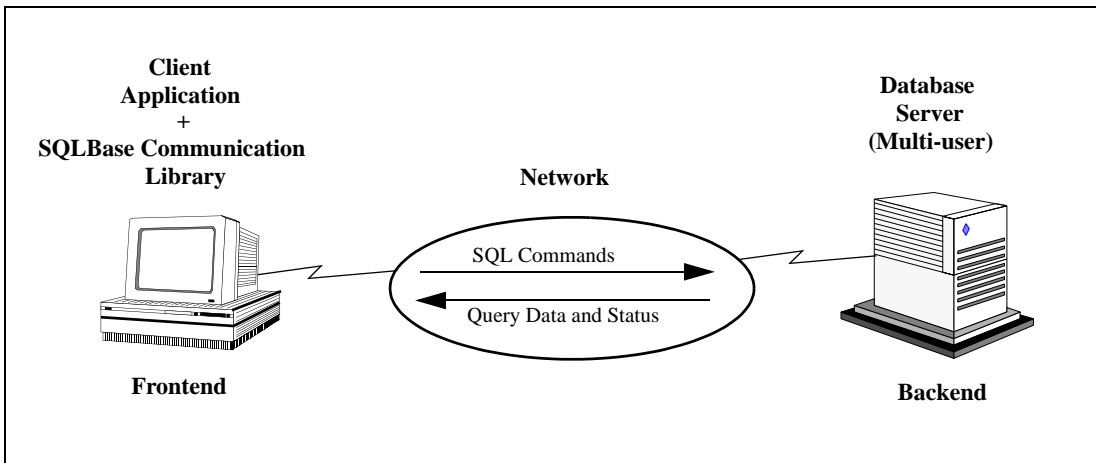
A database server and clients on a LAN

The difference from file servers

Database servers are different from file servers. File servers are responsible for storing, not processing, data files. Application processing is the client's responsibility and a client's request for data results in the file server sending an entire database across the network to the client. As the number of clients making requests for data increases, so does the likelihood of network performance degradation and bottlenecks.

In contrast, a relational database system has the flexibility and functionality necessary to work across networked computer systems. A client application uses SQL to talk to a database server. The client is responsible for executing the application and formatting a compact SQL command which it then sends across the network to the database server. The database server is responsible for parsing and executing the SQL command and sending data and a status code back to the client.

Unlike a file server, a database server sends only a part (subset) of the database back to the requesting PC. Most database servers use SQL because it is a convenient language for specifying logical subsets of data.



Cooperative processing: a client and server communicating via SQL

Benefits of distributed databases

The benefits of distributed databases are:

- **Location independence.** The location of a client application is independent of the location of the data.
- **Location transparency.** Users can access a database without having to know its location, and you can move a database with no affect on users or applications.
- **Incremental application growth.** You can upgrade an application by adding more clients or by purchasing a more powerful database server machine.
- **Site autonomy.** You can maintain each database separately from other databases and you can administer the shared data and allow programmers to support the applications.
- **You can distribute and secure data to match individual departmental needs.**
- **Hardware and software independence.** You can use hardware and software from different manufacturers, and you can match them to the requirements of your applications.
- **Distributed processing.** You can distribute the processing and storage of data among many computers.
- **Increased availability and reliability.** If a database server goes down, only its clients are affected, not your entire organization.

Chapter 2

Communication

Although it is not usually a DBA's responsibility to configure network software, you do need to know how SQLBase uses network resources and how to configure SQLBase for various communication options.

Overview

SQLBase software consists of both client and server components.

Client-side software is:

- Applications (like SQLTalk)
- SQL/API
- Communication libraries
- Database drivers and providers

Server-side software is:

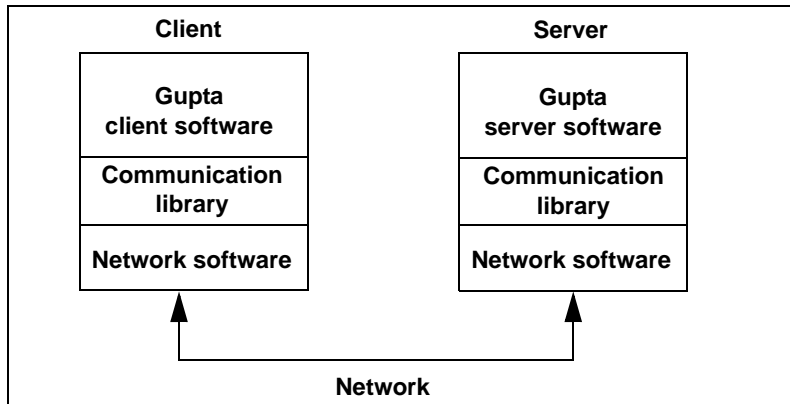
- Single and multi-user database servers
- Communication libraries

You can connect a client application to a local or remote SQLBase database. Local means that the client application and database reside on the same computer, while remote means that they reside on different computers.

You run an application and communication library on the client side, and a database server and communication library on the server side. The communication libraries on both sides enable Gupta client applications and SQLBase servers to communicate with each other over a network.

Communication libraries provide network protocol support and specify how an application and a database server communicate. Communication can take place between an application and a database server on the same computer (interprocess communication) or between an application and a database server across a network.

This diagram shows how SQLBase uses communication libraries to connect to the network software.

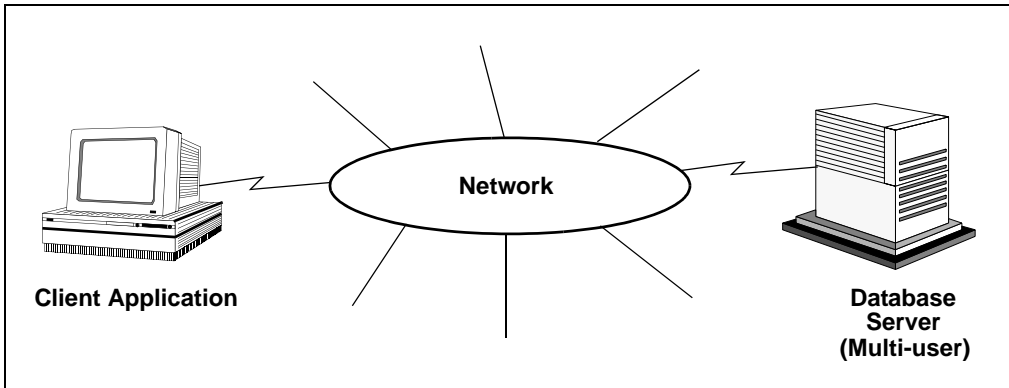


Local connection

In a local connection, you need only one computer. You run an application and a database server on the same machine. The database server accesses local databases only.

Remote connection

In a remote connection, you need at least two computers because a server and its clients are separate nodes on a network. The client and the server communicate with each other through a protocol such as TCP-IP.



A remote connection: a client and server communicating across a network.

The remote connection depends on your having installed network software on both the client and server computers. You should verify that the network software is loaded and working before attempting to connect a client and server.

Supported protocols

SQLBase supports these network protocols:

- Anonymous pipes
- TCP/IP
- SPX

See the *Client-to-server communication matrix* on page 2-4 to learn which clients can connect to which servers.

Protocols supported by platform

This table summarizes the platforms on which SQLBase runs and the protocols supported by SQLBase on each platform. .

Platform	Client or server	Protocols supported
Windows NT (4.0 and later), 2000, XP, and Server 2003P	Client	Anonymous pipes, SPX, TCP/IP
	Server	Anonymous pipes, TCP/IP
Windows 98/ME8	Client	Anonymous pipes, SPX, TCP/IP
	Server	Anonymous pipes, TCP/IP
NetWare	Client	SPX, TCP/IP
	Server	SPX, TCP/IP

Client-to-server communication matrix

The following table shows valid client and server communication, and their supported protocol options.

For example, a Windows NT client can communicate with a:

- NetWare server via either SPX or TCP/IP
- Windows 2000 server via anonymous pipes, or TCP/IP
- Windows 98/ME server via TCP/IP

Clients				
Servers	Platform	NetWare ¹	Windows NT/ 2000/2003/XP	Windows 98/ME
	Windows NT/2000/ 2003/XP	TCP/IP	Anonymous pipes TCP/IP	TCP/IP
	Windows 99/ME	TCP/IP	TCP/IP	Anonymous Pipes TCP/IP

NetWare	SPX TCP/IP	SPX TCP/IP	SPX TCP/IP
----------------	---------------	---------------	---------------

SQLBase client/server communication protocols by platform.

¹ You can build a SQL/API client application running on a NetWare client machine. Gupta provides a NetWare client communication library NLM to communicate between a NetWare SQL/API application and a SQLBase server.

Configuring SQL.INI

You configure communication through Connectivity Administrator or by editing SQL.INI directly with a text editor. Through SQL.INI, you control the communication libraries that are loaded, the protocols used, and settings for those protocols.

The file SQL.INI is structured in different sections. This basic structure is the same for both clients and servers (referred to here generically as “platform”). A client or server has a main section:

```
[platform]
parm1=setting1
parm2=setting2
```

The names of the platforms are:

Platform	Section name	Description
Clients	[win32client]	Windows 98, ME, NT, 2000 , XP, or Server 2003
	[nwclient]	NetWare (SQL/API application)
Windows 98/ ME and Windows NT/ 2000/2003/XP servers	[dbntsrv]	Unlimited user version
	[dbnt*sv]	1, 2, 5, 10, 25, or 50 user version; the asterisk in the actual name represents 1, 2, 5, 10, 25, or 50, depending on the version
NetWare 4.x servers	[dbnwsrv]	Unlimited user version
	[dbnw*sv]	2, 5, 10, 25, or 50 user version; the asterisk in the actual name represents 2, 5, 10, 25, or 50, depending on the version

A different section that is named by appending *.dll to the platform name specifies the communication libraries to load.

```
[platform.dll]
comdll=sqlapipe
comdll=sqlwsspx
```

If you specify two or more *comdll* keywords in the [**.dll*] section, then a client tries each communication library in turn until it finds the database. In the example above, the application first looks for the database locally because *sqlapipe.dll* is a local communication library. If it is unsuccessful, it then looks remotely, searching the network via SPX because *sqlwsspx.dll* is a SPX-specific communication library.

For most of the communication libraries, you optionally have a different section that specifies parameters for the protocol. The section is named by appending the *comdll* name (without the “sql”) to the platform name:

```
[platform.wsspx]
parm1=setting1
parm2=setting2
```

The statements below show the sections for a single user server:

```
[dbnt1sv]
...
dbname=ISLAND,SQLAPIPE,sqlws32
servername=server1,sqlapipe,sqlws32
```

```

...
[dbnt1sv.dll]
comdll=sqlapipe
comdll=sqlws32
...
[dbnt1sv.ws32]
...

```

The dbname statement specifies the protocols that a specific database uses for connections. The servername statement specifies protocols available for the server to use. In the example above, the server listens for connections using the anonymous pipes and TCP/IP protocols. The [dbnt1sv.dll] section specifies communication libraries to load when the server starts. In the example above, the server loads the communication libraries for anonymous pipes and TCP/IP when it starts.

The statements below show the sections for a client:

```

[win32client]
...
[win32client.dll]
comdll=sqlapipe
comdll=sqlwssp
comdll=sqlws32
...
[win32client.apipe]
...
[win32client.ws32]
...
[win32client.wssp]
...

```

The [winclient32.dll] section shows the communication libraries that the client loads when it first tries to connect to a database. In the example above, the client loads the communication libraries for anonymous pipes, SPX, and TCP/IP when it starts. The table below shows the name of the communication libraries and the associated comdll statements, the SQL.INI section, and the parameters.

Protocol	Platform	Library name	comdll statement	SQL.INI section	Parameters
TCP/IP	Windows 98, ME, NT, 2000, Server 2003, and XP	sqlws32.dll	comdll=sqlws32	[*.ws32]	listenport serverpath
	NetWare server	tlidll.nlm	comdll=sqltip	[*.tip]	

Protocol	Platform	Library name	comdll statement	SQL.INI section	Parameters
SPX	Windows 98, ME, NT, 2000, Server 2003, and XP with Novell NetWare client software	sqlspx32.dll	comdll=sqlspx32	[*.spx32]	connecttimeout serverpath
	Win32 client with Microsoft NetWare client software	sqlwsspx.dll	comdll=sqlwsspx	[*.wsspx]	
	NetWare server	tlispx.nlm	comdll=sqltsp	[*.tsp]	
Anonymous pipes	Win32 client and server	sqlapipe.dll	comdll=sqlapipe	[*.apipe]	serverpath

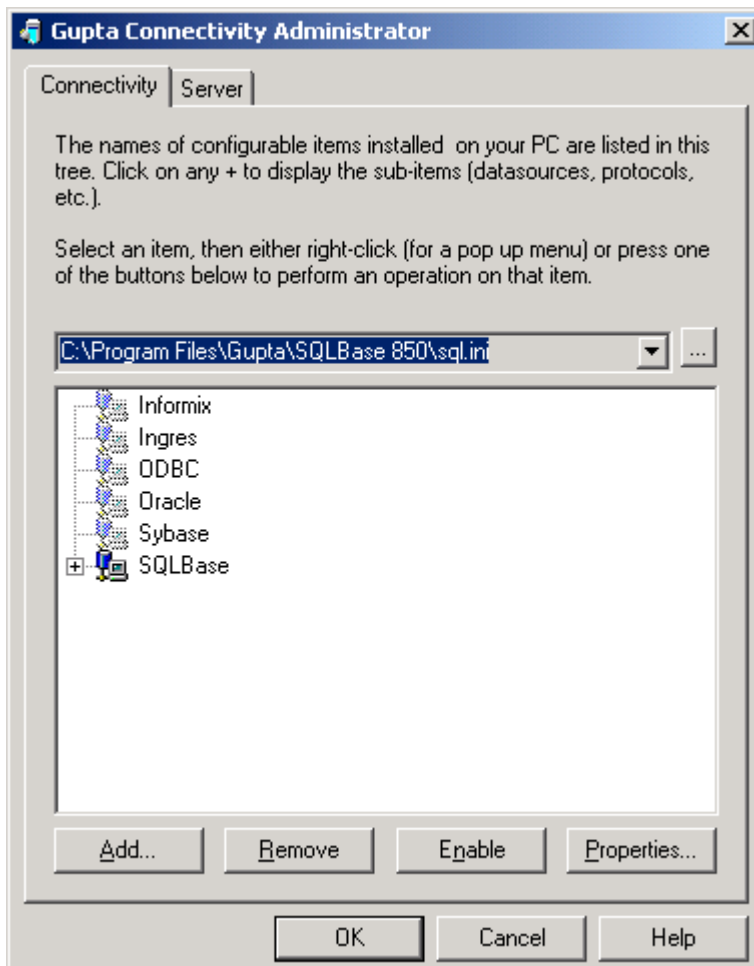
Connectivity Administrator

This section shows the basic steps to using Connectivity Administrator to configure communication on client and server platforms. For more about Connectivity Administrator, read the online help.

Important: You cannot use Connectivity Administrator to configure a NetWare server's SQL.INI file. You must edit the file with a text editor.

After you configure SQL.INI with Connectivity Administrator, you can open SQL.INI with a text editor to see the changes.

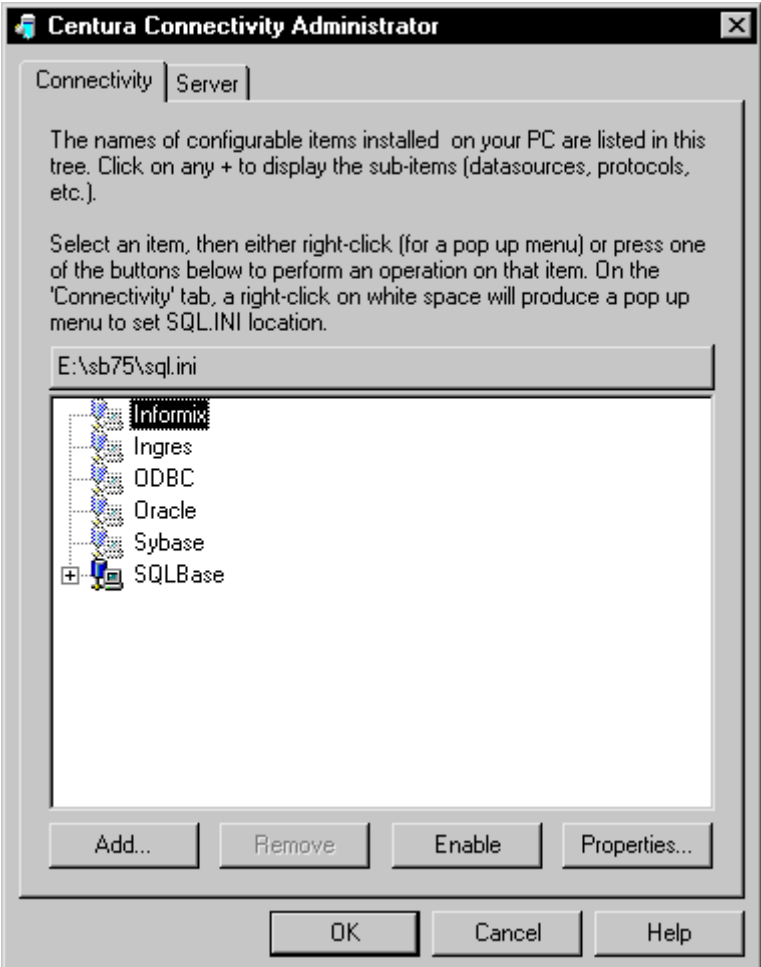
Beginning with version 8.5 Connectivity Administrator allows you to edit a configuration file by any name, in any location. This documentation still uses the name SQL.INI, but your actual file name is under your control. You can have multiple configuration files for multiple clients and servers, and you can tell Connectivity Administrator which file you wish to edit by typing or browsing a file name, as shown in the highlighted text below.



Configuring servers

1. Start Connectivity Administrator.
2. Click the Server tab, if it is not already displayed.
3. Click on one of the server nodes.
4. Expand the Listening Protocols branch.
5. Select a protocol and click the Enable button.

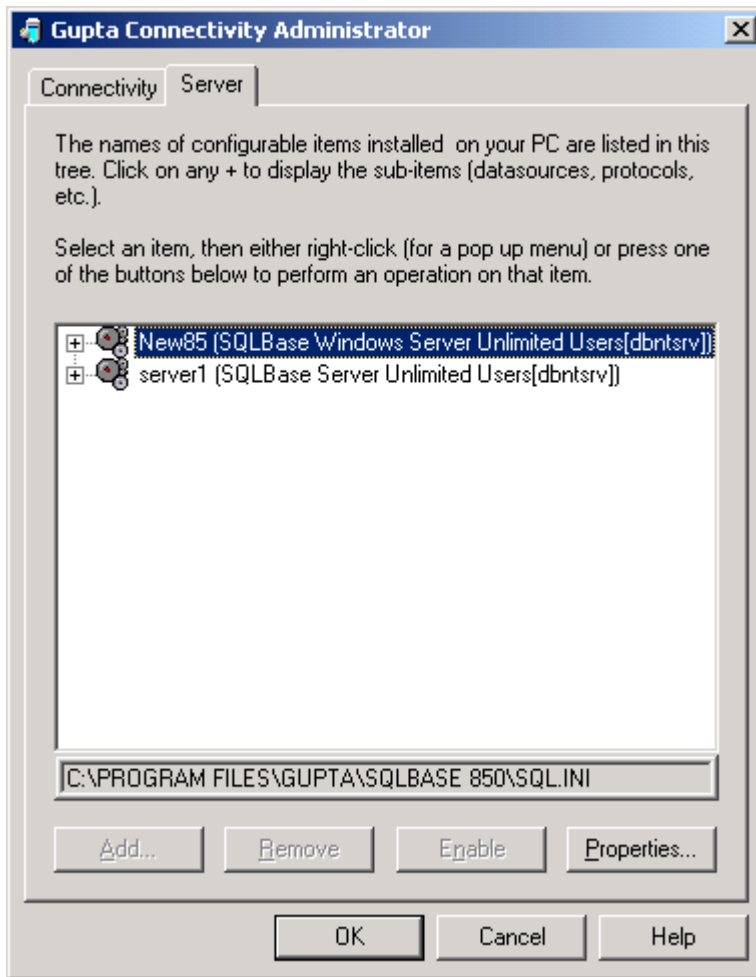
6. With the protocol selected, click the Properties button.



Note: When a protocol is selected, you can also right-click and select from a popup menu to enable a protocol and set its properties.



7. A dialog appears where you can set the protocol's settings.

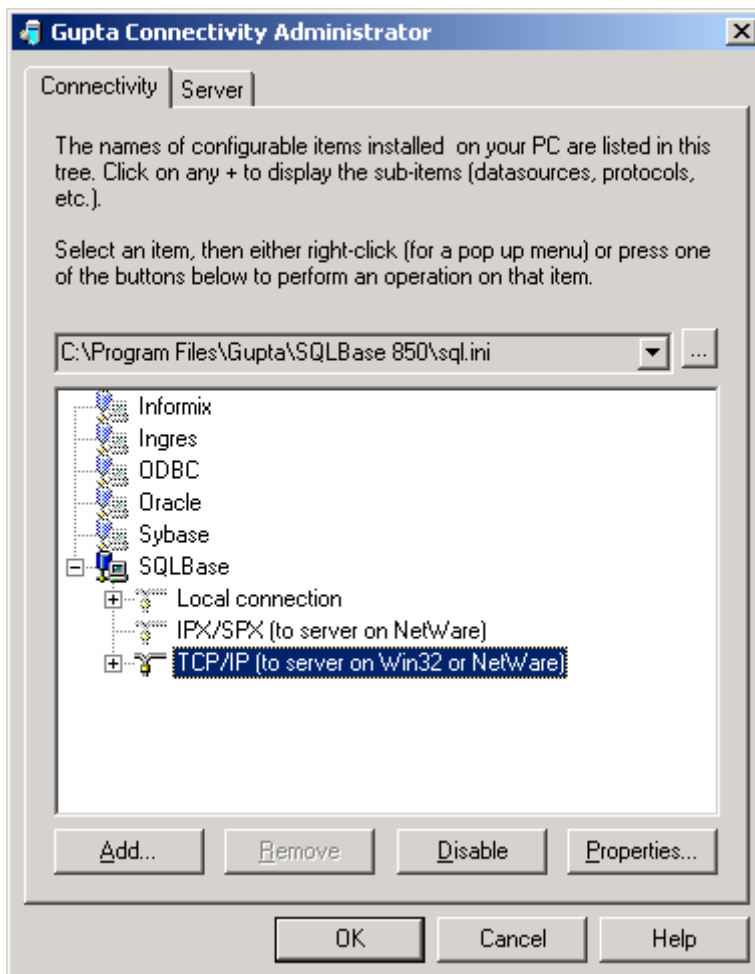


When multiple servers are installed on a single computer, and two or more servers are using the TCP/IP listening protocol, it will be necessary to alter the listening port for the servers, as illustrated above, so that only one is using the default port of 2155, and the others have their own unique port numbers. Remember that port 2156 is reserved for use by the SQLBase Resource Managers.

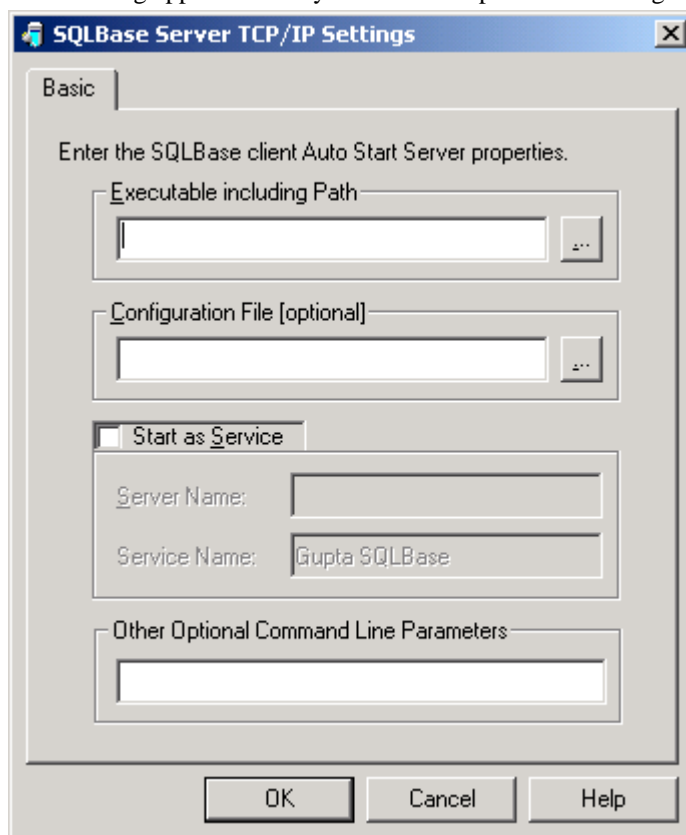
Configuring clients

1. Start Connectivity Administrator.
2. Click the Connectivity tab if it is not already displayed.

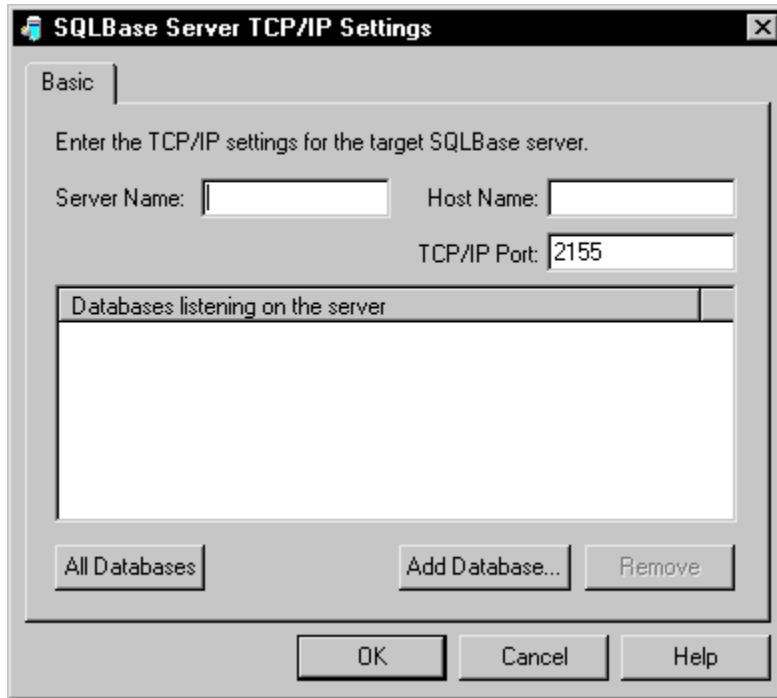
3. Expand SQLBase at the bottom of the list.
4. Select a protocol in the list and click Enable.
5. Click Properties.



6. A dialog appears where you can set the protocol's settings.



Note: After enabling TCP/IP on a client for the first time, you must click Add and fill in the dialog shown below. Connectivity Administrator uses this information to create a *serverpath* statement in SQL.INI.



Anonymous pipes

On Windows 98, ME, NT, 2000, Server 2003, and XP, an anonymous pipe enables clients and database servers on the same computer to transfer information back and forth as if they are reading and writing a file. Anonymous pipes cannot be used across a network.

TCP/IP

TCP/IP (Transmission Control Protocol/Internet Protocol) is a collection of networking protocols that have been used to construct the global Internet. TCP/IP is also widely used to build private networks called internets (spelled with a small "i") that may or may not be connected to the global Internet. An internet that is used exclusively by one organization is sometimes called an intranet.

On Windows 98, ME, NT, 2000, Server 2003, and XP, SQLBase uses winsock (Windows sockets) to communicate via TCP/IP.

If you enable TCP/IP on a client, you must also specify a serverpath statement in SQL.INI.

SPX

SPX (Sequenced Packet eXchange) is used in Novell Netware networks. SPX sits on top of IPX (Internetwork Packet eXchange).

On Windows 98/ME, you must have installed Novell's NetWare client software. On Windows NT/2000/XP/2003, you must have installed either Novell's NetWare client software or Microsoft's NetWare client software.

SNMP

This section describes how SQLBase uses the SNMP protocol. This protocol is only valid if you are using SQLConsole. SQLConsole uses SNMP to retrieve information from the SQLBase server.

What is SNMP?

The Simple Network Management Protocol (SNMP) is an open standard protocol that specifies how information is exchanged between devices on a network. The Internet Engineering Task Force (IETF) has adopted SNMP as the standard network protocol for managing network devices.

SNMP is implemented as an asynchronous protocol that is not dependent upon the network transport mechanism.

What does SNMP monitor?

SNMP is commonly used to retrieve information and manage a variety of network devices (such as hubs and bridges) from different vendors and on different platforms from a central location. For example, consider the following configuration:

- A bridge to a Novell Ethernet network with several Novell Netware file servers and a Windows/NT SQLBase database server also offering file services.
- Several hundred workstations.

Using SNMP, you can retrieve information both from the various network devices and also the servers on the network. If the devices networks have the same MIB (described on the following page), the information returned will be consistent and comparable regardless of vendor or platform. The management tool need only

understand SNMP and definitions about what information can be managed. This eliminates the need for several management tools in a multi-vendor environment.

In addition to monitoring hardware, the flexibility of SNMP allows you to monitor and manage virtually anything that communicates across a network, including software programs. For example, SQLConsole uses SNMP to retrieve information from the SQLBase server.

Components

There are three key components in an SNMP managed network:

- The MIB

The Management Information Base is the definition of what information is available, the description of that information, how it is accessed, and how it should be displayed. Objects in the MIB are identified by their Object Identifier (OID).

- The Agent

The agent provides information to satisfy SNMP requests for information pertaining to the managed device. The agent may also issue notification messages or traps to another device on the network, typically a management station. An SNMP agent is required on each managed device.

- The NMS

The Network Management Station is a network-connected workstation that can run software tools such as an SNMP-enabled custom application.

The NMS uses SNMP to issues requests, or *polls*, for information to managed devices on the network. Sometimes, the SNMP agent can volunteer this information on its own, without having to receive a request first.

The NMS uses SNMP to *poll* information from specific network device addresses. These SNMP requests contain both an OID (Object Identifier) that specifies which information is to be retrieved, and a network address specifying a device to receive the query. The NMS consults the MIB to establish what it can ask about and to determine how to package the request. Upon receiving an SNMP request, the agent on the target device consults its MIB and responds by returning the requested information if the agent is capable of satisfying the request.

How SQLBase and SQLConsole use SNMP

The SQLBase SNMP agent can provide information about a SQLBase server connected to a network. The information that can be monitored is described in the RDBMS MIB. This MIB is installed during SQLConsole installation.

The SQLBase agent

The SQLBase agent is a logical entity (process, thread, or extension to another program) running on a system that also runs a SQLBase database server. The agent is responsible for satisfying queries according to the definition of the RDBMS MIB.

The Network Management Station

The SQLBase implementation of SNMP provides an add-in program that extends the functionality of HP OpenView for Windows. This program provides the RDBMS MIB information specifically for the SQLBase Windows/NT/2000/XP SNMP Agent.

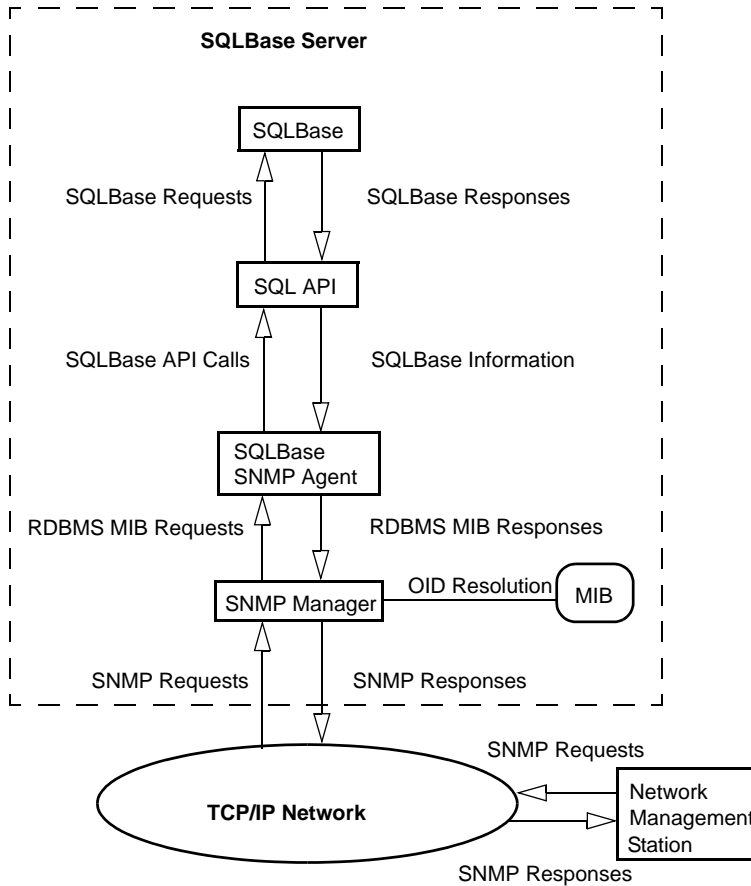
Any NMS communicating over a network using TCP/IP can query SQLBase for Windows NT (4.0 and later), 2000, XP, and Server 2003 SNMP Agent software.

The HP OpenView program provides the following features:

- Automatically recognizes the SQLBase SNMP Agent during discovery.
- Presents RDBMS information in a SQLBase specific manner.
- Launches SQLConsole from HP OpenView.

SNMP flow diagram

This diagram shows the general architecture and communication flow of the SQLBase SNMP Agent.



The SQLBase MIB allows you to monitor several servers at one network address. However, since only one instance of SQLBase can be executing on a given server at a time, information will only be available for one SQLBase database server. The SQLBase Database SNMP Agent only provides information about SQLBase; it does not provide information about databases from any other vendors.

Only the SQLBase server for Windows NT (4.0 and later), 2000, XP, and Server 2003 is supported by an SNMP agent. Those SNMP services use TCP/IP to communicate across the network. The server and the NMS must therefore have TCP/IP installed and operational. There is no requirement for other SQLBase client workstations to

use TCP/IP unless they are SNMP capable and wish to access the SQLBase SNMP agent.

Database drivers and providers

SQLBase provides industry-standard drivers and providers for client applications to use when communicating with SQLBase. In such cases the client need not be concerned about the specifics of other communication components, only about the correct use of the drivers and providers themselves. These drivers and providers are explained in detail in the book *Guide to Connecting to SQLBase*. The supported platforms are:

ODBC database driver

JDBC database driver

OLE DB data provider

.NET data provider

COM+ and the SQLBase Resource Manager

Beginning with version 8.0, SQLBase transactions can be integrated into larger COM+ transactions through the use of the SQLBase Resource Manager (SQLBRM). SQLBRM is the component that enables COM+ transactions against SQLBase. It is notified by the Microsoft Distributed Transaction Coordinator (DTC) when a COM+ component initiates a transaction against the SQLBase server. When an application running under COM+ issues a command to commit or abort a transaction, DTC notifies SQLBRM, which causes the command to be executed in SQLBase. If the SQLBase server fails during a transaction, SQLBRM notifies DTC, and later plays back and commits the transaction when the SQLBase server restarts.

SQLBRM.EXE is installed by default as a Windows service. You can run the executable directly instead, but ordinarily you would use it as a service. It takes three parameters:

```
SQLBRM /ListenPort=2156 /LogDir=c:\windows\temp /Autostart=yes
```

The values shown are the default values used when SQLBRM is installed as a service (presuming that “windows” is the directory where your Windows operating system is installed.)

The Autostart parameter controls whether SQLBRM will start SQLBase if it's not already running. If SQLBRM is a service, it will start SQLBase only as a service, presuming that SQLBase has been configured to allow this.

Configuring servers and clients

To use SQLBRM with COM+, you must specify the SQLMPIPE protocol in the section of SQL.INI that relates to the database engine:

```
[dbntsrv]
dbname=ISLAND,sqlmpipe
servername=server1,sqlmpipe
...
[dbntsrv.dll]
comdll=sqlmpipe
...
```

The SQL.INI settings related to clients require that you use SQLWS32 as the communication protocol when running against database engines using SQLMPIPE:

```
[win32client.dll]
comdll=sqlws32
[win32client.ws32]
serverpath=server1,yourhostname,2156/*
...
```

Note the use of 2156 as the default listen port for SQLBRM, as contrasted with the default listen port of 2155 for the ordinary SQLBase TCP/IP protocol. This port number must match the port number used when SQLBRM is started.

Shutting down SQLBase Resource Manager

When running as a service, Resource Manager can be shut down from the Windows Services administrator, or from SQLBase Management Console (described below). When Resource Manager receives a request to shut down, it waits until any existing users have disconnected from SQLBase.

Chapter 3

Configuration: SQL.INI and the Registry

This chapter describes:

- The structure and organization of Gupta's configuration file (*SQL.INI*) and the configuration keywords
- The registry entries that SQLBase uses

SQLBase configuration file

SQLBase saves configuration information in a file with the default name of SQL.INI. All Gupta client and server software products read this file when they start.

Beginning with version 8.5, SQLBase permits multiple installations to exist and operate simultaneously. One of the effects of this feature is that you may rename SQL.INI to any other name you wish, and may store it in any path on disk that you wish. Note that this feature is optional.

Note: For purposes of documentation, we will always refer to the SQLBase configuration file as **SQL.INI**, even though the actual name of the file on your computer may be different.

The name and the location of the configuration file for a particular installation are stored in the registry, under these two keys:

```
HKEY_CURRENT_USER\  
Software\Gupta\SQLBase\servername\ConfigFileName  
HKEY_CURRENT_USER\  
Software\Gupta\SQLBase\servername\ConfigFilesDirectory
```

The term *servername* refers to whatever name you assigned the database server upon installation. Server name can also be changed using the SQLBase Management Console or the Connectivity Administrator.

Finding the configuration file

Earlier versions of SQLBase used several methods of locating SQL.INI. Version 8.5 is more straightforward. Each server executable (for example, dbntsrv.exe) accepts a command-line argument that argument contains the file name (and, optionally, the path) of the configuration file. The name might be something other than SQL.INI.

If the command-line argument is left blank when the executable starts, it is presumed that the configuration file is named SQL.INI and that it resides in the same directory as the executable itself.

How clients find SQL.INI

SQLBase clients and pre-7.0 versions of SQLBase server look for the configuration file first in the directory you specify in the SQLBASE environment variable. If not found there, the search order for this file is:

1. Current directory.
2. \sqlbase directory on the current drive.
3. Root directory on the current drive.
4. Directories specified by the PATH environment variable.

5. The directory of the executable.
6. The Windows system directory.
7. The Windows directory.

The configuration file name is presumed to be SQL.INI. Client applications can also call API function *sqliniEx* when beginning a database session. This function accepts a parameter containing the name (and, optionally, the path) of the configuration file to use.

Editing SQL.INI

You can edit SQL.INI in two ways:

- You can use the Gupta Connectivity Administrator utility if you are running Windows 98, ME, NT, 2000, XP, or Server 2003. For examples of using Connectivity Administrator, read *Connectivity Administrator* on page 2-8 and *Tutorial* on page 7-28. Note that the Connectivity Administrator online help contains topics for all the keywords documented in this chapter.
- You can use your choice of text editor to edit SQL.INI directly.

Gupta recommends that you use Connectivity Administrator for editing.

SQL.INI format

SQL.INI is divided into sections, each starting with a section identifier enclosed in square brackets. Each section identifier corresponds to a particular client or server platform.

In most cases, section identifiers are based on executable or DLL names. For example, the section corresponding to SQLBase Server for Windows NT (4.0 and later), 2000, XP, and Server 2003, whose executable name is *dbntsrv.exe*, is [dbntsrv].

For a list of SQL.INI section names for each client and server platform, read *Configuring SQL.INI* on page 2-5.

Communications

For information about configuring SQL.INI for communication, read *Chapter 2, Communication*.

dbdfault section

There is one section which does not fit into either of the two cases (platforms and communications) already mentioned.

Section	Description
[dbdfault]	Saves the <i>defaultdatabase</i> , <i>defaultuser</i> , and <i>defaultpassword</i> settings for Win32 clients.

Configuration entries

Each section of SQL.INI contains keywords and values:

```
[section identifier]
keyword1=value
keyword2=value1,value2,. . .
keyword3=value1,. . .
```

The number of values differs for each keyword.

For example:

```
[dbntrsrv]
cache=300
servername=server1,sqlnpipe,sqlntnbi
dbname=demo1,sqlnpipe,sqlntnbi
```

Keywords are described later in this chapter.

Command line keywords

You can also set the value of some keywords when you start SQLBase. Specify a keyword and its value on the command line to override a SQL.INI configuration entry.

For example, the following command starts SQLBase Server for Windows and specifies the database home directory (*dbdir*) as *d:\test\db*. This overrides a *dbdir* configuration entry in SQL.INI.

```
dbntrsrv dbdir=d:\test\db
```

You can specify more than one keyword on the command line, each separated by a space, for example:

```
dbntrsrv dbdir=d:\test\db log=d:\test\dblog
```

Syntax rules

Use the following rules when creating configuration entries:

- Specify keywords in any order.

For example, configuring the *servername* and *dbname* keywords in this order:

```
servername=server1,sqlnpipe,sqlntnbi
dbname=demol,sqlnpipe,sqlntnbi
```

is functionally equivalent to configuring them in this order:

```
dbname=demol,sqlnpipe,sqlntnbi
servername=server1,sqlnpipe,sqlntnbi
```

- List keyword values in the order required by the keyword.

For example, you would get an error if you tried to change the order of:

```
servername=server1,sqlnpipe,sqlntnbi
```

to:

```
servername=sqlnpipe,sqlntnbi,server1
```

because the *servername* keyword expects the value of the server name to be listed first, followed by one or more communication libraries.

- Use a comma (,) to separate multiple keyword values.

For example:

```
dbname=demol,sqlnpipe,sqlntnbi
```

Note that the exception to this rule is the *dbdir* keyword which uses semicolons to separate path names.

- Use a semi-colon (;) to start a comment line. The rest of the line is ignored.

For example:

```
; This specifies the name of a database as well as the
; protocols on which the server listens for clients
; making connection requests to that database.
dbname=demol,sqlnpipe,sqlntnbi
```

- Limit each entry to the length of one line.

SQLBase gets an error if you allowed a comment line, for example, to span multiple lines:

```
; This comment line is too long to fit on one line,
and ends up spanning multiple lines...
```

- Do not place spaces before or after keyword values.

For example, SQLBase gets an error if you put spaces between the *dbname* keyword values:

```
dbname=demol, sqlnpipe, sqlntnbi
```

You must eliminate spaces between keyword values:

```
dbname=demo1,sqlnpipe,sqlntnbi
```

- Keywords are case-independent.

For example:

```
servername=server1,sqlnpipe,sqlntnbi
```

is functionally equivalent to:

```
SERVERNAME=server1,sqlnpipe,sqlntnbi
```

When do keyword changes take effect?

Most SQL.INI keywords can be changed at any time, and their changed behavior will take effect the next time the associated database server or client is started. There is one exception to this rule: keyword *country*. This keyword relate to the National Language Support feature in SQLBase. If you use National Language Support, a new starter database (start.dbs) is created when you select a country. The settings of this keyword are associated with the new starter database (for example, germany.dbs) at that time. Any database that is then created on that server will use the starter database as its basis, and the remembered value of *country* will be permanently incorporated in that new database throughout its physical lifetime, regardless of how the SQL.INI keywords are altered after start.dbs is created.

You must not change the *country* setting after database creation, either in the server portion of SQL.INI or the client portion. Doing so would lead to error messages about “invalid codepages”.

If you decide that you wish to safely use a different setting for *country*, you must unload the affected databases, follow all the steps outlined in *Building a database with NLS* on page 10-9, then load the contents back into those newly built databases.

Keyword list

The following table lists and provides a brief description of each configuration keyword:

Keyword	Brief Description
<i>ansijoinsyntax</i>	Indicates whether SQL99 ANSI syntax can be used in queries for purposes of joining tables.
<i>audit</i>	Specifies the path and directory name for the audit file managed in the SQL.INI file by the START AUDIT and STOP AUDIT commands.

Keyword	Brief Description
<i>autostartserverpath</i>	Specifies the full name and location of the SQLBase server executable, for use with the WS32, APIPE, and MPIPE protocols.
<i>batchpriority</i>	Allows you to change the priority level SQLBase sets for itself. You can lower the priority from the default high priority to “normal” Windows priority level.
<i>cache</i>	Specifies the number of pages in the database cache. Page size is 1 kilobyte. Minimum value is 15, maximum is 1,000,000.
<i>centurydefaultmode</i>	Changes the two-digit default century values that SQLBase stores in a database.
<i>characteraset</i>	Identifies a file that specifies different values for the ASCII character set.
<i>clientcheck</i>	Instructs the server to send the client a RECEIVE message upon receipt of a request.
<i>clientname</i>	Specifies the name that the server displays in the CLIENT NODE field on its Server Status display when a client connects to a database.
<i>clientruntimedir</i>	Specifies the location of SQLBase client binary files.
<i>cmdtimeout</i>	Specifies the amount of time to wait for a command to complete execution.
<i>comdll</i>	Specifies the communication libraries to load.
<i>commitserver</i>	Enables a server to act as the commit server for distributed transactions.
<i>connectpausesticks</i>	Number of seconds that the client waits before deciding that an attempt to connect to the server has failed.
<i>connecttimeout</i>	Specifies the total time allowed, in seconds, for the client to repeatedly attempt to connect to the server before a timeout message is issued.
<i>country</i>	Instructs SQLBase to use the settings in the specified section of the <i>country.sql</i> file.
<i>dbdir</i>	Specifies the drives, paths, and directory names for the home database directories.
<i>dbname</i>	Specifies a database name and, optionally, communication libraries that clients use to access the database.
<i>dbwin</i>	Determines the status of the Databases subwindow.
<i>defaultdatabase</i>	Specifies the default database name, overriding the default of DEMO.
<i>defaultpassword</i>	Specifies the default password, overriding the default of SYSADM.
<i>defaultuser</i>	Specifies the default user name, overriding the default of SYSADM.
<i>defaultwrite</i>	Controls whether changes to the values of the <i>defaultdatabase</i> , <i>defaultpassword</i> , or <i>defaultuser</i> keywords are written to SQL.INI.

Keyword	Brief Description
<i>directsap</i>	Allows you to specify a Service Advertising Protocol (SAP) engine for SQLBase servers running under NetWare 4.x.
<i>disablelogspacecheck</i>	Allows you disable checking for log file space availability before a new log file is opened.
<i>displevel</i>	Specifies the level of information (0-4) displayed on the Process Activity window.
<i>errorfile</i>	Specifies a file that contains entries to translate standard SQLBase return codes into user-defined return codes.
<i>extdll</i>	Specifies the pre-loading of DLLs at server start-up time.
<i>fileaccess</i>	Prevents user access to select SQL/API functions.
<i>groupcommit</i>	Specifies the maximum number of <i>commits</i> that SQLBase groups together before physically writing them to disk.
<i>groupcommitdelay</i>	Specifies the maximum number of system ticks that SQLBase waits before performing a <i>commit</i> .
<i>inmessage</i>	Sets the size of the input message buffer.
<i>insertioncontext</i>	Specifies where in the NDS Tree SQLBase server and database objects reside.
<i>listenport</i>	Identifies the port number on which a server listens for connections.
<i>listenretry</i>	Specifies the number of times a server should retry a <i>listen</i> when an attempt to listen on the network fails.
<i>locks</i>	Specifies the maximum number of lock entries to allocate.
<i>locktimeout</i>	Specifies how long to wait when acquiring a lock.
<i>log</i>	Writes all the messages that appear on the server's Process Activity display to a specified file.
<i>logdir</i>	Redirects the transaction logs to the specified drive and directory.
<i>logfileprealloc</i>	Enables and disables transaction log file preallocation.
<i>mainwin</i>	Determines the size and position of the main server window.
<i>maxnestinglevel</i>	Specifies the maximum nesting level for recursing stored procedures.
<i>ndsloginid</i>	Specifies the userid to log into the NDS tree.
<i>ndsloginpassword</i>	Sets a password for the NDS user ID that is used to log into the NDS tree.

Keyword	Brief Description
<i>negotiateapi</i>	Enables secure 7.1 routers to connect to servers that do not support transmission security.
<i>netcheck</i>	Enables and disables a checksum feature that detects transmission errors between a client and a server.
<i>netchecktype</i>	Specifies the algorithm SQLBase uses when <i>netcheck</i> is enabled.
<i>netlog</i>	Invokes a diagnostic server utility that records database messages to a specified log file.
<i>numconnectionecb</i>	Adjusts the number of connection Event Control Blocks (ECBs).
<i>numlistenecb</i>	Used to adjust the number of listen Event Control Blocks (ECBs).
<i>nwadvertisemode</i>	Specifies what mode to use when advertising SQLBase servers and databases on the network.
<i>optimizefirstfetch</i>	Sets the optimization mode for a result set.
<i>optimizerlevel</i>	Determines the optimizing techniques that SQLBase uses for all clients that connect to a server.
<i>oracleouterjoin</i>	Turns on and off Oracle-style outer join processing.
<i>osavgwindow</i>	Sets the number of samples of the operating system statistics to keep for determining average value.
<i>ossamplerate</i>	Sets the frequency at which operating system statistics are gathered.
<i>outmessage</i>	Sets the size of the output message buffer.
<i>partitions</i>	Enables access to partitioned databases
<i>password</i>	Sets a password for the server.
<i>preferrednameservice</i>	Specifies a Novell name service to use.
<i>procwin</i>	Determine the status of the Process Activity subwindow.
<i>readonly</i>	Enables and disables read-only mode for all databases on a server.
<i>searchcontext</i>	Specifies what part of the NDS tree that an NDS client requires to search for the SQLBase server name and installed database names.
<i>secureapi</i>	For clients, sets the encryption level used in messages between the client and server. For servers, sets the minimum level of encryption of messages to which the server responds.
<i>servername</i>	Specifies a name for a multi-user server.

Keyword	Brief Description
<i>servername</i>	Specifies the names of one or more LAN Manager file servers to which a database server connects.
<i>serverpath</i>	Indicates the path that a client using the TCP/IP protocol uses to locate SQLBase servers.
<i>showmaindbname</i>	Enables or disables the display of the MAIN database when you query the server for a list of database names.
<i>silentmode</i>	Turns the display for a multi-user server on and off.
<i>sortcache</i>	Specifies the number of cache pages to use for sorting. Maximum value is 65536.
<i>statwin</i>	Determines the status of the Server Status subwindow.
<i>syswin</i>	Determines the status of the System Activity subwindow.
<i>tempdir</i>	Specifies the directory where SQLBase places temporary files.
<i>threadmode</i>	Specifies whether to use native NetWare threads or SQLBase threads.
<i>threadstacksize</i>	Specifies the stack size.
<i>timeout</i>	Specifies the time period that the server waits for a client to make a request.
<i>timestamps</i>	Specifies the timestamp setting.
<i>timezone</i>	Sets the value of SYSTIMEZONE, a SQLBase keyword that returns the time zone as an interval of Greenwich Mean Time.
<i>users</i>	Specifies the maximum number of client applications (or processes) that can connect to a server simultaneously.
<i>workalloc</i>	Specifies the basic allocation unit of a work space.
<i>worklimit</i>	Specifies a maximum memory limitation for SQL commands.

Keyword descriptions

The configuration keyword descriptions that follow contain:

- Purpose
- Whether configuration is mandatory, recommended, or optional
- Where to configure the keyword within SQL.INI
- Default value, if any
- Example

If you incorrectly configure a keyword, SQLBase returns an error when you try to start the appropriate program.

ansijoinsyntax

Determines whether SQL99 ANSI join syntax is permitted in queries. A value of “1” means that ANSI syntax is permitted. The default value is “0” (not permitted). See the SQL Language Reference manual for detailed information on use of this syntax.

audit

Specifies the path and directory name for the audit file managed in the SQL.INI file by the START AUDIT and STOP AUDIT commands. For details on database auditing, read *Database auditing* on page 4-18. For descriptions of the parameter entries that are automatically provided when you start an audit, read the START AUDIT command documentation in the *SQLBase SQL Language Reference*.

autostartserverpath

Specifies the path and name of the SQLBase server executable. The WS32, APIPE, and MPIPE protocols use this information. Values containing embedded spaces must be enclosed by double quotes.

By specifying this keyword, you make client applications capable of starting the SQLBase server, if they wish to connect and the server is not already running. Note that the server started in this way can only run as a program, even if the server has been configured to be capable of running as a Windows service.

(Starting SQLBase as a Windows service can be accomplished from SQLBase Management Console, the Windows Services window, or by using the NET START command from the command line.)

batchpriority

Determines what priority will be assigned to the SQLBase server task within the Windows operating system. Choices are 0 and 1. 0 causes SQLBase to demand a larger share of CPU time and resources than ordinary Windows applications. 1 causes SQLBase to have the same priority as an ordinary Windows application.

default value The default value is 0.

cache

Specifies the number of pages in the database cache. The cache buffers database pages in memory. The larger the cache, the less the disk input and output. In other words, as you increase the value of the *cache* setting, disk access is reduced.

Note that the combined values of *cache* and *sortcache* cannot exceed the memory resources of your system.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value The default cache size is 2000 pages.

The minimum is 15 pages and the maximum is one million pages.

You should experiment with various *cache* values to determine the best setting. The best setting is that which maximizes the performance of your applications.

Example To set the number of pages in cache to 200, specify:

```
cache=200
```

Note: Page size is fixed at 1K.

centurydefaultmode

This keyword provides year 2000 (Y2K) support. By default, SQLBase always stores 2-digit century values as the current century. To change the default setting, specify 1 (one) as the value for the *centurydefaultmode* keyword. When set to 1, SQLBase applies the algorithm reflected in the following table to determine whether the year is in the current, previous, or, next century.

When last 2-digits of current year are:	When 2-digit entry is 0-49	When 2-digit entry is 50-99
0-49	The input date is in the current century	The input date is in the previous century
50-99	The input date is in the next century	The input date is in the current century

Note: Enabling the 2-digit century is a SQLBase feature and has no impact on connectivity routers. If you are using a Gupta developed application or a SQL/API application against a non-SQLBase database, read the database documentation for information on how it determines year/century values.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Example

Assume the current year is 1996:

If 05 is entered, the computed date is 2005
 If 89 is entered, the computed date is 1989

- Assume current year is 2014:
 If 05 is entered, the computed date is 2005
 If 34 is entered, the computed date is 2034
 If 97 is entered, the computed date is 1997
- Assume current year is 2065:
 If 05 is entered, the computed date is 2105
 If 70 is entered, the computed date is 2070

characteraset

Identifies a file that specifies different values for the ASCII character set.

This is useful for non-English speaking countries where characters in the ASCII character set have different hexadecimal values than those same characters in the U.S. ASCII character set. For example, in the U.S., 0x41 is an 'A'. In another country, 0x41 might be a 'G'.

When you insert data into a database, the specified file is used by SQLBase to translate non-U.S. ASCII characters to U.S. ASCII characters

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Example

For example, assume you specify:

```
characteraset=abc.chr
```

and the file *abc.chr* contains the line:

```
0x09, 0x10, 0x45, ...
```

SQLBase makes the following translation:

From... To...

0x09	0x00
0x10	0x01

0x45 0x02
...

SQLBase reverses the process when you fetch the data back, making the following translation:

From... To...

0x00 0x09
0x01 0x10
0x02 0x45
...

clientcheck

Tells SQLBase to send the client a RECEIVE message upon receipt of a request.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value By default, *clientcheck* is off (0). When SQLBase has finished executing a command, it issues a SEND request to the client with the results of the command. If the SEND message is successful, the server then issues a RECEIVE request and waits to receive another command.

Setting *clientcheck* on (1) instructs SQLBase to issue a RECEIVE request before beginning execution of the command, not after it finishes executing the command. Doing so allows SQLBase to detect a situation where the client session is dropped or a cancel request is made during command processing.

Example To set *clientcheck* on, specify:

 clientcheck=1

clientname

Specifies the name that a server displays in the CLIENT NODE field on its Server Status display when a client connects to a database.

This keyword is optional. Configure this keyword in any client communication library section of SQL.INI.

Default value Based on the platform, SQLBase has various ways of generating the CLIENT NODE value displayed on the Server Status screen:

- For Windows CLIENT NODE displays a randomly-generated number.
- For a client whose SQL.INI includes the *timebased* configuration keyword, CLIENT NODE displays a timestamp.

Specifying a value for the *clientname* keyword overrides this default. The value of *clientname* can be from 1 to 12 characters in length.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Example To name a client 'workstation1', specify:

```
clientname=workstation1
```

clientruntimedir

Specifies the path where client binary files and .SQL files can be found. This keyword is used in the WIN32CLIENT section of SQL.INI. Values containing embedded spaces must be enclosed by double quotes.

Example `clientruntimedir=c:\program files\gupta\my8.5`

cmdtimeout

Specifies the amount of time (in seconds) the server should wait for a SELECT, INSERT, UPDATE, or DELETE command to complete execution. After the specified time has elapsed, SQLBase rolls back the command.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value Valid values are:

1 to 43,200 (1 second to 12 hours)

0 (infinity; wait forever; default)

Example To set a limit of one minute, specify:

```
cmdtimeout=60
```

comdll

Specifies the communication libraries (DLLs) to load. Communication libraries specify the mechanism that provides communications between a client and a server. Communication can be across a network or interprocess on a single computer.

For details about configuring this keyword, read *Chapter 2, Communication*.

This keyword is mandatory. Configure this keyword in the server communications or client router communications section of SQL.INI.

You can specify one or more communication libraries in a client or server's SQL.INI. A server can listen on multiple protocols simultaneously. A client can only use one protocol at a time, so a client attempts to make a connection with each communication library in turn until it finds one that works.

For example, if your network supports both TCP/IP and SPX but a client uses TCP/IP more often, the *comdll* keyword entries should be in this order:

```
[win32client.dll]
```

```
comdll=sqlws32  
comdll=sqlwsspx
```

This ensures that SQLBase first tries to make the connection via Windows TCP/IP, and if that fails it tries to make the connection via SPX.

Example If a Windows client and server communicate via TCP/IP, specify:

```
comdll=sqlws32
```

This tells the server to listen for the client's connection request on the TCP/IP protocol and tells the client to use this same protocol to make the connection.

commitserver

Enables a SQLBase server to act as the commit server for distributed transactions.

The two-phase commit protocol uses a commit server that logs information about each distributed transaction. The commit server performs the following functions:

- Assigns a global ID to the transaction. This global ID distinguishes a distributed transaction from a regular transaction.
- Starts and ends a distributed transaction.
- Commits or rolls back a distributed transaction.
- Logs the status and various stages of a distributed transaction.
- Assists in recovery in case of a crash.

One of the servers participating in the distributed transaction must have *commitserver* enabled. If both servers are enabled, SQLBase arbitrarily selects one to be the commit server.

This keyword is optional. Configure this keyword in any multi-user server section of SQL.INI.

Default value By default, *commitserver* is not enabled (0).

Example To enable commit server capability, specify:

```
commitserver=1
```

connectpauseticks

Specifies the amount of time (in seconds) that the client should wait for an individual attempt to connect to the server. After the specified time has elapsed, the client tries again or, if *connecttimeout* time is exceeded, a failure message is issued. Specified in the client section.

Default value Default value is 10 seconds:

connecttimeout

Specifies the amount of time (in seconds) the client should wait for repeated attempts to connect to the server. After the specified time has elapsed, a failure message is issued.

This keyword is optional. Configure this keyword in the client section of SQL.INI.

Default value Default value is 5 seconds:

country

Instructs SQLBase to use the settings in the specified section of the *country.sql* file.

This keyword is optional. Configure in any of the following SQL.INI sections:

- A Windows client
- Any client router or server section

For more information, read *Chapter 10, National Language Support*.

Note: The behavior of the *country* keyword is different from most SQL.INI settings. For more, read *When do keyword changes take effect?* on page 3-6.

Default value SQLBase supports English as the default language.

Language. SQLBase also supports many international languages including those spoken in Europe and Asia. You specify information that enables SQLBase to support another language in the *country.sql* file. This file contains a section for each country that SQLBase supports, and the section name is the country name as coded for the *country* keyword in SQL.INI.

To support French:

```
country=france
```

The matching entry in the *country.sql* file would look like this:

```
[france]
keyword1=value
keyword2=value
```

ASCII versus EBCDIC. The *country* keyword also enables you to choose between using ASCII or EBCDIC formatted data on your server machine. Using EBCDIC mode enhances the testing of programs designed to run on a mainframe, and eases data exchange between the mainframe and the PC.

The *country* keyword corresponds to an entry in the *country.sql* file where you can configure an entry in the [ebcdic] section to define the EBCDIC sort sequence.

Example To support German, specify:

```
country=germany
```

dbdir

Specifies the drives, paths, and directory names for the home database directories. For partitioned databases, *dbdir* indicates the location of the MAIN database when you create the first partitioned database.

A home database directory contains a subdirectory which in turn contains a database file and, optionally, one or more log files. The name of the subdirectory must be the same as the database file name without the extension. For example, the home database directory for the *demo.dbs* database is the directory that contains the DEMO subdirectory.

Note that the home database directories are specified with a drive letter and path name. If you omit the drive letter, the current drive is assumed. The exception to this would be a URL referencing a network path, such as:

```
dbdir=\\SomeOtherMachine\\SomeVolume\\Gupta
```

For operating systems that have environment variables, you can set the DBDIR variable to tell SQLBase the path to database files.

SQLBase version 8.5 searches for databases using the following locations, in this order:

1. The directory specified by *dbdir* in SQL.INI (if set).
2. The directory specified by %DBDIR% in the environment (if set).
3. For all ordinary databases, if a *dbdir* specification existed but the database was not found, the search stops here. If no *dbdir* specification existed, the older search sequence (described below) will be used. If searching for START.DBS, the directory containing the server executable is searched at this point.

Versions of SQLBase prior to 8.5 used the following search sequence:

1. The directory specified by *dbdir* in SQL.INI (if set)
2. The directory specified by %DBDIR% in the environment (if set)
3. The current directory
4. The directory \sqlbase on the current drive
5. The root of the current drive
6. The directory of the server executable
7. The Windows system directory
8. The Windows directory

SQLBase creates a new database in the first valid directory specified by the above search locations.

If SQL.INI contains two *dbdir* keywords, the second setting overrides the first.

You can change the setting of *dbdir* in SQL.INI with the SQLTalk command SET SERVER DBDIR and the *sqlset* API function (with the SQLPDBD parameter). If you change the setting of *dbdir* through either of these ways, it does not affect a server that is currently running. The server does not see the new settings until the next time it starts.

This keyword is recommended. Configure this keyword in the server section of SQL.INI.

Default value Current directory.

Example You can specify a path with one or more directories. The maximum length of the *dbdir* value is 260 characters. The maximum number of characters allowed in a *dbdir* directory path (excluding the filename which has a limit of eight characters and a three-character extension) is 246. This includes the drive letter, colon, leading backslash, and terminating null character.

Semicolons and commas cannot be part of any path name. Semicolons are used as path delimiters, while commas are used as special characters to separate each parameter of a keyword in SQL.INI. A directory name enclosed in single quotes can contain an embedded single quote which is indicated by TWO single quotes (for example: D:\'"test"\

The syntax is the same as for the DOS path command:

```
dbdir=d:\Gupta;d:\devel;d:\test
```

Note: Characters reserved in the Windows environment cannot be used in directory or file names; for example: <> : “ / \ | . Words reserved in the Windows environment cannot be used as file names; for example, *con* or *aux*.

For SQLBase Server for NetWare, be sure to include the volume (db:):

```
dbdir=db:\Gupta
```

dbname

This keyword has slightly different uses depending on whether you specify it in the server's or the client's SQL.INI.

Client configuration

Specifies a database name and communications libraries that the client should use to connect to that database. The client can connect to the database using *only* those communications libraries.

If you do not specify a *dbname* keyword, the client tries to connect to a database using the communications libraries specified by the *comdll* keyword in the [application.dll] section, in the order in which you listed the *comdll* keywords.

This keyword is optional. Configure this keyword in the client section of SQL.INI.

Default value This keyword has no default value.

Example To connect to the test database using TCP/IP, specify:

```
dbname=test,sqlws32
```

You can also specify a wildcard character (*) for the database name:

```
[application]  
dbname=*,sqlws32
```

The wildcard matches any database name that the user specifies. The above example directs the client to connect to the database using TCP/IP.

Server configuration

Specifies a database name on the server. This name must be unique on the network. The server listens on the network for client requests to access this database and when a client attempts to connect to the database, the server helps establish the connection.

Make sure that for each *dbname* entry, there exists a corresponding database. Otherwise, the network does not recognize the database.

The following SQL commands and SQL/API functions update the *dbname* setting:

SQL Command	SQL/API Function
CREATE DATABASE	sqlcre
DROP DATABASE	sqldel
INSTALL DATABASE	sqlind
DEINSTALL DATABASE	sqlded

This keyword is recommended. Configure this keyword in any multi-user server section of SQL.INI.

Default value The value of this keyword defaults to demo.

Example To identify the test database, specify:

```
dbname=test
```

Remember that there must be a *test.dbs* file in the TEST subdirectory of the home database directory or in the path defined by the database home directory *dbdir*.

You can optionally specify one or more communication library names after the database name. For example, SQLBase Server for Windows NT (4.0 and later), 2000, XP, and Server 2003 supports anonymous pipes, SPX, and TCP/IP. The following configuration entry directs the server to listen on the network for connection requests to the database using *all* of those communication libraries:

```
dbname=test,sqlpipe,sqlws32,sqlwsspx
```

Conversely, the following configuration entry directs the server to listen on the network using *only* the TCP/IP communication library:

```
dbname=test,sqlws32
```

Only those client requests to access the test database that are sent via TCP/IP will be successful.

Remember that any communication libraries that you specify for the *dbname* configuration entry, must also be specified for the *comdll* configuration entry.

dbwin

Determines the status of the Databases subwindow.

The syntax for this window keywords is identical to that for the *mainwin* keyword, with the addition of a *closed* status. This indicates that the given subwindow was closed when you last used the **Save** option of the **File** menu.

This keyword is optional. This keyword is configured automatically in the server's GUI section (for example, [dbntsrv.gui]) when you use the **Save Settings** option in the server's **File** menu. Do not set a value for this keyword manually.

Example

The syntax is:

```
dbwin=status,x,y,cx,cy
```

The following values are valid for these settings:

- **Status**
Indicates the status of the window when you last used the **File**, **Save** option. Valid settings are: NORM for normal window status, MAX if the window is maximized, MIN if the window is minimized, or CLOSED if the window is closed.
- **x, y**
These values represent the position of the window. For Windows NT (4.0 and later), 2000, XP, and Server 2003, this is the window's top-left corner.
- **cx,cy**

These values represent the size of the window. For Windows NT (4.0 and later), 2000, XP, and Server 2003, these are the coordinates of the window's lower-right corner. These values are system-dependent; do not manually modify or copy them from one system to another.

defaultdatabase

Specifies the default database name, overriding the default of DEMO.

This keyword is optional. Configure this keyword in the Windows default-specific section [dbdfault], or in a Windows client's [win32client] section.

Default value This keyword has no default value.

Example To set the default database name to 'test', specify:

```
defaultdatabase=test
```

defaultpassword

Specifies the default password, overriding the default of SYSADM.

This keyword is optional. Configure this keyword in the Windows default-specific section [dbdfault] or in a Windows client's [win32client] section.

Default value This keyword has no default value.

Example To set the default password to 'secret', specify:

```
defaultpassword=secret
```

defaultuser

Specifies the default user name, overriding the default of SYSADM.

This keyword is optional. Configure this keyword in the Windows default-specific section [dbdfault] or in a Windows client's [win32client] section.

Default value This keyword has no default value.

Example To set the default user name to 'admin', specify:

```
defaultuser=admin
```

defaultwrite

Controls whether changes to the values of the *defaultdatabase*, *defaultpassword*, or *defaultuser* keywords are written to SQL.INI.

This keyword is optional. Configure this keyword in the Windows default-specific section [dbdfault] or in a Windows client's [win32client] section.

Default value Valid values are 0 (off) and 1 (on). The default is 1.

Example To set *defaultwrite* off, specify:

```
defaultwrite=0
```

This specifies that changes to the *defaultdatabase*, *defaultpassword*, and *defaultuser* keywords are in effect for the current session only and are not written to SQL.INI.

When *defaultwrite* is on, changes to the default keywords are remembered by SQLBase for the current and all future sessions. As well, the specified changes are written to SQL.INI.

directsap

Allows you to specify whether to use Novell's Service Advertising Protocol (SAP) engine or the alternative Gupta SAP engine for the SQLBase Server for NetWare 4.x.

Under NetWare, the Novell SAP engine has a maximum of 20 SAPs. This means that if you have more than 15 databases installed and you try to connect to additional databases, the connection fails. Using the Gupta SAP engine removes this limitation.

The syntax of this keyword is:

```
directsap=n
```

where n is either 0 (use Novell's SAP engine) or 1 (use Gupta's SAP engine).

This keyword is optional. Configure this keyword in the SPX section for the SQLBase Server for NetWare 4.x, for example, [dbnwsrv.spx] for the unlimited version).

Example To use Gupta's SAP engine, specify:

```
directsap=1
```

Default value The default is 0.

disablelogspacecheck

Disables SQLBase from checking log file space availability before a new log file is opened. SQLBase opens a new log file when the current one has reached its specified maximum size. If the log file space is inadequate to ensure database integrity should a system failure occur, SQLBase selectively rolls back any transaction that is causing excessive logging. SQLBase issues an error message that informs you the transaction has been rolled back, asks you to check for free disk space, and warns of imminent system shut down if the roll back does not recover log disk space.

By default, LOGBACKUP is off, so that SQLBase deletes log files without backups as soon as they are not needed to perform transaction rollback or crash recovery. This prevents log files from accumulating and filling up the disk. When *disablelogspacecheck* is enabled, be sure to set LOGBACKUP to off; otherwise, SQLBase is unable to attempt recovery of log disk space. To set LOGBACKUP, use the SQLTalk SET LOGBACKUP command or the *sqlset* API function.

This keyword is optional. Configure this keyword in the server section of SQL.INI. This keyword is not applicable to Windows 3.x and on any partitioned database.

Example	The syntax is: disablelogspacecheck=n
Default value	When this keyword is not declared, the default behavior is to perform checking for available log file space. Otherwise, to specify the default behavior, the value is 0. To disable checking, the value is 1.
Example	The syntax is: disablelogspacecheck=n To disable checking for available log file space, specify: disablelogspacecheck=1

displevel

Specifies the level of information (0-4) displayed on the Process Activity window. Set the display level with either the Level menu on the GUI server, the SQLTalk SET PRINTLEVEL command, or the *sqlset* SQL/API call in conjunction with the SQLPPLV parameter.

This keyword is optional. This keyword is configured automatically in the server's GUI section (for example, [dbntrsvr.gui]) when you use the **Save Settings** option in the server's **File** menu. Do not set a value for this keyword manually.

Example	The syntax is: displevel=n
---------	-----------------------------------

errorfile

Specifies a file that contains entries to translate standard SQLBase return codes into user-defined return codes.

This keyword is optional. Configure this keyword in the Windows default-specific section [dbdfault], or in a Windows client's [win32client] section.

Default value	This keyword does not have a default value.
---------------	---

Example	The syntax is: errorfile= <i>filename</i> where <i>filename</i> includes the full path.
---------	---

The maximum length of the *errorfile* value is 260 characters. The maximum number of characters allowed in an *errorfile* directory path (excluding the filename which has a limit of eight characters and a three-character extension) is 246. This includes the drive letter, colon, leading backslash, and terminating null character.

Semicolons and commas cannot be part of the pathname. A directory name enclosed in single quotes can contain an embedded single quote which is indicated by TWO single quotes (for example: D:\'"test'"').

Note: Characters reserved in the Windows environment cannot be used in directory or file names; for example: <> : “ / \ |. Words reserved in the Windows environment cannot be used as file names; for example, *con* or *aux*.

The file contains entries for error code translation in the form:

```
sbrcd,udrcd
```

where *sbrcd* is a SQLBase return code found in *error.sql*, and *udrcd* is a user-defined return code. The *sbrcd* value must be a positive integer; the *udrcd* can be a positive or negative integer. There can be no white space between the values or after the comma.

The client application converts the *sbrcd* value to the *udrcd* value using the *sqltec* API function. For example, SQLBase returns a value of '1' to indicate an end-of-fetch condition, while DB2 returns a value of '100'. If you want an application to convert all SQLBase return codes of '1' to '100', the entry in the errorfile would look like this:

```
1,100
```

When your application calls the *sqltec* function, if the SQLBase return code doesn't exist, SQLBase returns a non-zero return code that means that the translation did not occur.

To force translation to occur, you can create a global translation entry using the asterisk (*) character and a generic return code (like '999'). For example, assume we created an errorfile of SQLBase return codes and corresponding DB2 return codes. For those SQLBase return codes that have no corresponding DB2 return code, you can force the application to return the generic return code '999' with the following entry:

```
*,999
```

extdll

Specifies the pre-loading of DLLs at server startup time. Because of the enormous overhead involved in making calls to the Microsoft API function load library, (especially in the case of large DLLs or DLLs that cause more DLLs to be loaded), pre-loading DLLs saves overhead and guarantees that the DLL is loaded as long as the server is running.

Note: If you have a DLL that uses global variables that can be accessed from different functions or from multiple invocations of a function, you must load the DLL at server start up.

If you want to pre-load more than one DLL, you can specify the parameter multiple times within the server section. Each DLL entry must be on a separate line.

This keyword is optional. Add this keyword to a Windows server section of SQL.INI.

Default value By default, SQLBase loads the DLL at function call time by calling the Microsoft API function load library.

Example The syntax is:

```
EXTDLL=dllname
```

where *dllname* is a full path or the operating system uses the path environment variable to locate *dllname*. If the DLL name is not qualified, the operating system uses the path environment variable to locate the DLL.

The maximum length of the *extdll* value is 260 characters. The maximum number of characters allowed in the *extdll* directory path (excluding the filename which has a limit of eight characters and a three-character extension) is 246. This includes the drive letter, colon, leading backslash, and terminating null character.

Semicolons and commas cannot be part of any pathname. Semicolons are used as path delimiters, while commas are used as special characters to separate each parameter of a keyword in SQL.INI. A directory name enclosed in single quotes can contain an embedded single quote which is indicated by TWO single quotes (for example: D:\'test'\).

Note: Characters reserved in the Windows environment cannot be used in directory or file names; for example: <> : “ / \ | . Words reserved in the Windows environment cannot be used as file names; for example, *con* or *aux*.

fileaccess

Allows or disallows a user ability to access and modify files on a remote server machine. Use the this keyword to prevent access to any type of remote file.

Setting this keyword to zero (0) prevents a client SQL/API application from calling these functions:

- sqlfgt
- sqlfpt
- sqlmop
- sqlmdl

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value The default for this keyword is 1 which allows remote access to server files via the SQL/API.

Example The syntax is:

```
fileaccess=0
```

Setting *fileaccess*=0 disallows a client application from reading or writing to a remote server file.

groupcommit

Specifies the maximum number of COMMITs that SQLBase groups together before physically writing them to disk. *Commits* are performed when:

- The number of grouped *commits* exceeds the value of *groupcommit*, or
- The number of ticks that have elapsed since the last *commit* is greater than the value of *groupcommitdelay*. Read the *groupcommitdelay* description in the next section for more information on the *groupcommitdelay* keyword.

SQLBase tries to economize resources and increase transaction rates by grouping COMMITs from several transactions into one transaction log entry, and performing them all in one physical write. It does this by pausing for a fraction of a moment in anticipation of other incoming COMMIT requests. SQLBase does not defer COMMITs; it always writes a COMMIT to disk before returning control to the user.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value The default value is 1.

Example To set the maximum to 5, specify:

```
groupcommit=5
```

groupcommitdelay

Specifies the maximum number of system ticks that SQLBase waits before performing *commits*. The duration of a system tick is platform-dependent, but is approximately 1/20 of a second.

Commits are performed when:

- The number of grouped *commits* exceeds the value of *groupcommit*, or
- The number of ticks that have elapsed since the last *commit* is greater than the value of *groupcommitdelay*.

Read the *groupcommit* description in the previous section for more information on the *groupcommit* keyword.

This keyword is optional. Configure this keyword in the single-user or multi-user server section of SQL.INI.

Default value The default value is 1 tick.

Example To set the groupcommitdelay to 20, specify:

```
groupcommitdelay=20
```

inmessage

Sets the size (in bytes) of the input message buffer. The input message buffer holds input to the client application (such as the result of a query).

There is one input message buffer per connected cursor on the client computer. The database server maintains a buffer that is the size of the largest input message buffer on the client computer. The server builds a message in its buffer, then sends it across the network to the client's input message buffer. It is called an input message buffer because it is input from the client's perspective.

This keyword is optional. Configure this keyword in the 32-bit client router section [win32client.spx32] of SQL.INI, except for the Windows NT platform.

Default value The default size is 2000 bytes. Despite the *inmessage* value, SQLBase dynamically allocates more space if necessary.

When fetching data, SQLBase compacts as many rows as possible into the input message buffer. Each fetch reads the next row from the buffer until all have been read. At this point, SQLBase transparently fetches another buffer-ful of rows, depending on the isolation level.

A large input message buffer can improve performance when reading long varchar data and while fetching data because it reduces the number of network messages between the client and server. A large buffer can have a negative impact on concurrency, however, because any row currently in the buffer can have a shared lock on it (depending on the isolation level) which prevents other users from changing that row. Read the *sqlsil* documentation in the *SQLBase Application Programming Interface Reference* for more information about how each isolation level uses the input message buffer.

Example To set the size of the input message buffer to 5,000 bytes, specify:

```
inmessage=5000
```

insertioncontext

Specifies where in the NDS tree the SQLBase server and database objects reside.

For details about configuring this keyword, read *Insertion context* on page 11-2.

This keyword is optional. For SQLBase Server for NetWare 4.x, configure this keyword in the server section of SQL.INI.

Example To specify that the SQLBase server and database objects reside in a part of the directory tree identified as CN=SERVERxx, enter:

```
[dbnwsrv]
insertioncontext=CN=SERVERxx.OU=SQLBASE.OU=Menlo.O=Gupta
```

Default value If you do not specify this keyword, the default context of the user specified by the *ndsloginid* keyword is used for insertion.

listenport

Identifies the port number on which a server listens for connections. On a given machine, a SQLBase server obtains exclusive use of the identified port. You must ensure that the port chosen does not conflict with other applications on the same machine.

If you specify this keyword, client applications connecting to this server must use the same port number with the *serverpath* keyword in their corresponding client section. If you specify *listenport* in a client section, you will receive an error.

Contact your network administrator if you do not know what port value to use for this keyword.

This keyword is optional. Configure this keyword in a server's TCP/IP section of SQL.INI. For Windows 98, ME, NT, 2000, Server 2003, and XP platforms, this is [*servername*.ws32], for example, [dbnwsrv.ws32]. For the NetWare platform, this is [*servername*.tip], for example, [dbnserver.tip].

Default value The default port number is 2155.

The syntax of this keyword is:

```
listenport=n
```

where *n* is a positive integer not greater than 32767.

Example To set the port Number to 3000, enter the following line:

```
listenport=3000
```

listenretry

Specifies the number of times a server should retry a *listen* when an attempt to listen on the network fails.

This keyword is optional. Configure this keyword in a multi-user server's communications section (for example, [dbntsrv.ntnbi]) of SQL.INI.

Default value The default value of this keyword is 3.

Example To set the number of retries to 5, specify:

```
listenretry=5
```

locks

Specifies the maximum number of lock entries to allocate. SQLBase allocates lock entries dynamically (in groups of 100) on an as-needed basis.

Specify this keyword only when you are trying to protect against a run-away application that is accessing too much data in one transaction.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value The default value of this keyword is 0, which means that there is no limit on the number of locks allocated; as many lock entries can be allocated as memory permits.

Example To limit lock entry allocation to 500, specify:

```
locks=500
```

locktimeout

Specifies how long to wait when acquiring a lock. If an application cannot acquire a lock within the specified period of time, SQLBase returns a timeout error.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value The default value of this keyword is 300 seconds for all servers but SQLBase single-user for Windows. The default for SQLBase single-user for Windows is 2 seconds.

Example To set the timeout period to 2 minutes, specify:

```
locktimeout=120
```

log

Writes all the messages that appear on the server's Process Activity display to a specified file.

This keyword is equivalent to using SQLTalk's SET ACTIVITYLOG command.

These messages are helpful when debugging, but they are not meant to be logged in a production environment. Logging incurs overhead, and the log file can become large very quickly if your site is an active one.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value This keyword does not have a default value.

Example To write the messages to a file called *debug.txt*, specify:

```
log=d:\test\debug.txt
```

The maximum length of the *log* value is 260 characters. The maximum number of characters allowed in an *log* directory path (excluding the filename which has a limit of eight characters and a three-character extension) is 246. This includes the drive letter, colon, leading backslash, and terminating null character.

Semicolons and commas cannot be part of the pathname. A directory name enclosed in single quotes can contain an embedded single quote which is indicated by TWO single quotes (for example: D:\'test'').

Note: Characters reserved in the Windows environment cannot be used in directory or file names; for example: <> : “ / \ |. Words reserved in the Windows environment cannot be used as file names; for example, *con* or *aux*.

logdir

Redirects the transaction logs to the specified drive and directory.

When SQLBase creates log files, it appends the database name to the specified path name to ensure that log files for different databases not placed in the same directory. If the directory specified in logdir does not exist, SQLBase first creates the directory, then redirects the log files to that directory.

This keyword is recommended. Configure this keyword in the server section of SQL.INI.

Default value By default, SQLBase creates transaction log files in the home database directory. Redirecting the transaction log files has two benefits:

- If the database and its logs are on separate disk drives, higher transaction throughput is possible because of less disk arm movement and the ability to do parallel disk accesses.
- If the database and log files are on different disk drives, a media failure does not destroy both the logs and the database.

Example	<p>To redirect log files to the d:\test directory, specify:</p> <pre>logdir=d:\test</pre> <p>The maximum length of the <i>logdir</i> value is 260 characters. The maximum number of characters allowed in an <i>logdir</i> directory path (excluding the filename which has a limit of eight characters and a three-character extension) is 246. This includes the drive letter, colon, leading backslash, and terminating null character.</p> <p>Semicolons and commas cannot be part of the pathname. A directory name enclosed in single quotes can contain an embedded single quote which is indicated by TWO single quotes (for example: D:\'test').</p> <hr/> <p>Note: Characters reserved in the Windows environment cannot be used in directory or file names; for example: <> : “ / \ . Words reserved in the Windows environment cannot be used as file names; for example, <i>con</i> or <i>aux</i>.</p> <hr/>
---------	--

logfileprealloc

	<p>Enables and disables transaction log file preallocation.</p> <p>This keyword is optional. Configure this keyword in the server section of SQL.INI.</p>
Default value	<p>By default, <i>logfileprealloc</i> is off (0) and the current log file grows in increments of 10 percent of its current size. This uses space conservatively, but can lead to a fragmented log file which can affect performance.</p> <p>Setting <i>logfileprealloc</i> on (1) means that SQLBase creates log files full size (preallocated) and they do not grow.</p>
Example	<p>To enable log file preallocation, set <i>logfileprealloc</i>=1 in SQL.INI:</p> <pre>logfileprealloc=1</pre> <p>Start SQLTalk and issue a SET LOGFILESIZE command.</p> <p>From SQLTalk, issue a RELEASE LOG command. This forces SQLBase to start a new transaction log which it creates with the size you specified in the previous step.</p>

mainwin

	<p>Determines the position and size of the main server window.</p> <p>This keyword is optional. This keyword is configured automatically in the server’s GUI section (for example, [dbntsrv.gui]) when you use the Save Settings option in the server’s File menu. Do not set a value for this keyword manually.</p>
Example	<p>The syntax is:</p> <pre>MAINWIN=status,x,y,cx,cy</pre>

The following values are valid for these settings:

- **Status**
NORM for normal window status, MAX if the window is maximized, and MIN if the window is minimized.
- **x, y**
These values represent the position of the window. For Windows NT (4.0 and later), 2000, XP, and Server 2003, this is the window's top-left corner.
- **cx,cy**
These values represent the size of the window. For Windows NT (4.0 and later), 2000, XP, and Server 2003, these are the coordinates of the window's lower-right corner. These values are system-dependent; do not manually modify or copy them from one system to another.

maxnestinglevel

Specifies the maximum nesting level for recursing stored procedures.

This keyword is optional. This keyword is configured automatically in the server's GUI section (for example, [dbntrsvr.gui]) when you use the **Save Settings** option in the server's **File** menu. Do not set a value for this keyword manually.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value The minimum value for this keyword is 1; maximum value is 16. Maximum value is dependent on the limit of memory.

Example The syntax is:

```
maxnestinglevel=1
```

ndsloginid

Specifies the userid required to log into the NDS tree for adding, deleting, or changing the SQLBase server and database object entries.

For details about configuring this keyword, read *NDS logon ID and password* on page 11-5.

This keyword is mandatory only if NDS is used for advertising SQLBase servers and databases. Otherwise this keyword is ignored.

For the SQLBase Server for NetWare 4.x, configure this keyword in the server section of SQL.INI.

Default value By default, this keyword uses the login context of the NDS user.

Example To set the NDS login ID to **admin**, specify:

```
ndsloginid=admin
```

ndsloginpassword

Sets a password for the NDS userid required to log into the NDS Tree for adding, deleting, or changing the SQLBase server and database object entries. For details about configuring this keyword, read *NDS logon ID and password* on page 11-5.

This keyword is mandatory only if NDS is used for advertising SQLBase servers and databases. Otherwise this keyword is ignored. Configure this keyword in the server section of SQL.INI.

Default value By default, this keyword uses the login context of the NDS user.

Example To set the NDS login ID's password to **admin**, specify:

```
ndsloginpassword=admin
```

negotiateapi

You must specify this keyword if your client must connect to unsecure/foreign servers, and you need secure communications with the editions of SQLBase featuring encryption. If you set this keyword to 1, the SQL/API will first use the security specified by the *secureapi* keyword. Then if the connection fails, the client will attempt to connect with no security. Note: This setting reduces performance during connections if negotiation is required. For details about configuring this keyword, read *Transmission security* on page 7-25.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value By default, *negotiateapi* is off (0).

Example To enable *negotiateapi*, specify:

```
negotiateapi=1
```

netcheck

Enables and disables a checksum feature that detects transmission errors between a client and a server. To use this feature, both the client and the server must enable *netcheck*.

The checksum feature is enabled on the server side when you bring the server up. When a client attempts to connect to a database on the server, it requests the network check. If the server confirms the request, then the checksum feature is enabled on the client side.

It is the network's responsibility to ensure data integrity. However, the checksum feature can offer additional protection. In cases where the network is unable to detect

a problem (such as a memory problem in the communication card), the checksum feature can prevent corrupted data from being passed to either the client or the server.

If an error is detected by either the client or the server, the detector tries to transmit the data again. If an error from the server is detected and the process activity level is 2 or more, SQLBase displays a message indicating its intention to re-transmit. If the error re-occurs, SQLBase returns an error code to the client, aborts the transmission, and displays a message on the server's Process Activity screen. The corrupted data is not processed.

Most networks such as Ethernet, Token-Ring, and ARCNET check for errors using CRC-16 or CRC-32 algorithms. However, an error can occur on the path that the transmitted data takes from the network board to other components in the computer or in network gateways. It is these types of errors that this feature detects.

Using this network checksum feature can decrease performance by as much as 5 to 10 percent.

This keyword is optional. Configure this keyword in any multi-user server section of SQL.INI.

Default value By default, *netcheck* is off (0).

Example To enable *netcheck*, specify:

```
netcheck=1
```

Remember to add this entry to the SQL.INI files on both the server and the client.

netchecktype

Specifies the algorithm SQLBase uses when *netcheck* is enabled. Configure this keyword only when you enable *netcheck*.

This keyword is optional. Configure this keyword in any client router or multi-user server section of SQL.INI.

Default value By default, checksum (0) is enabled.

Example To switch to CRC/16, specify:

```
netchecktype=1
```

netlog

Invokes a diagnostic server utility that records database messages to a specified log file. This utility logs all messages that pass between a server and clients communicating over a network using SPX.

Do not use *netlog* unless instructed to do so by Gupta's Technical Support staff. You must start the netlog utility on a new database or one whose state you know to be

uncorrupted. Do not start netlog on a database that you know to have a problem. Starting netlog when a database already has a problem will not help, because the purpose of netlog is to show how the problem happened in the first place. Once you have recorded the problem, the technical staff at Gupta can play the netlog file back to reproduce the problem.

Syntax. The syntax of the keyword is:

```
netlog=filename[,force]
```

where ‘filename’ is the name of a file to which the messages will be written and ‘force’ is an optional parameter.

You should use the FORCE parameter for multi-user environments. If you do not use FORCE, the order of your transactions may not be the same in the log replay as they were for the original run.

This keyword is optional. Configure this keyword in any multi-user server section of SQL.INI.

Default value By default, the netlog utility is off.

Example To invoke the netlog utility and write to a file called *dbmsgs.log*:

```
netlog=dbmsgs.log
```

In fatal error situations, you may be asked by your technical support staff contact to add the FORCE option:

```
netlog=dbmsgs.log,force
```

nwadvertisemode

Specifies what mode to use when advertising SQLBase servers and databases on the network.

For details about configuring this keyword, read *Advertising mode* on page 11-5.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Example To instruct SQLBase to use the *nwadvertisemode* keyword to advertise the server and its databases only on the NDS tree, specify:

```
nwadvertiseMode= 2
```

Default value The default value for *nwadvertisemode* is 1.

optimizefirstfetch

Used to set the optimization method for the first fetch. When set, the keyword instructs the optimizer to pick a query execution plan that takes the least amount of time to fetch the first row of the result set.

The valid values for this keyword are 0 and 1. When *optimizefirstfetch* is set to 0, SQLBase optimizes the time it takes to return the entire result set. When *optimizefirstfetch* is set to 1, SQLBase optimizes the time it takes to return the first row.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Example To instruct SQLBase to use the *optimizefirstfetch* keyword to optimize the time of the first fetch, specify:

```
optimizefirstfetch=1
```

The setting applies for all queries being compiled for the current cursor, unless a query is compiled under a scope of a cursor with a different optimization mode.

Default value The default value for *optimizefirstfetch* is 0.

optimizerlevel

Determines the optimizing techniques that SQLBase uses for all clients that connect to a server.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value By default, *optimizerlevel* is 2 which causes SQLBase to use current optimizing techniques.

If you set *optimizerlevel* to 1, SQLBase uses old optimizing techniques. This setting lets you fall back on old optimizing techniques after upgrading to newer versions of SQLBase. If you discover better performance of a query when *optimizerlevel* is set to 1, you should report it to Gupta's Technical Support team. Be sure not to include compilation time in the comparison of settings 1 and 2.

Example To tell SQLBase to use old optimizing techniques:

```
optimizerlevel=1
```

You can override the *optimizerlevel* setting for a particular cursor with SQLTalk's SET OPTIMIZERLEVEL command or the *sqlset* API function with the SQLPOPL parameter.

oracleouterjoin

Turns on and off Oracle-style outer join processing.

Oracle's outer join implementation differs from the ANSI and industry standard implementation. To paraphrase the ANSI standard, the correct semantics of an outer join are to display all the rows of one table that meet the specified constraints on that

table, regardless of the constraints on the other table. For example, assume two tables (A and B) with the following rows:

Table A (a int)	Table B (b int)
1	1
2	2
3	3
4	
5	

If you issue the following SQL command:

```
SELECT a, b FROM A, B WHERE A.a = B.b (+) AND B.b IS NULL;
```

the ANSI result is:

Table A (a int)	Table B (b int)
1	
2	
3	
4	
5	

Assuming the same two tables and the same SQL command, the correct result for Oracle is:

Table A (a int)	Table B (b int)
4	
5	

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value

By default, *oracleouterjoin* is off (0) and SQLBase implements outer joins according to the ANSI standard.

Example

To direct SQLBase to process outer joins the Oracle way, specify:

```
oracleouterjoin=1
```

If you set *oracleouterjoin* on (1), you receive the Oracle result shown above.

osavgwindow

Specifies the number of samples of the CPU % utilization value to keep for determining the average value. You can specify a window size of 1 to 255.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value

By default, *osavgwindow* is 1.

Example

To specify a window size of 25:

```
osavgwindow=1
```

You can override the *ossamplerate* with the *sqlset* API function with the SQLPAWS parameter.

ossamplerate

Specifies the frequency at which operating system statistics (CPU % utilization) are gathered. You can specify a setting of 0 to 255 seconds.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value By default, *ossamplerate* is zero (0), which disables the gathering of CPU statistics.

Example To tell SQLBase to gather statistics every 10 seconds:

```
ossamplerate=1
```

You can override the *ossamplerate* with the *sqlset* API function with the SQLPOSR parameter.

outmessage

Sets the size (in bytes) of the output message buffer. The output message buffer holds output from the client application (such as a SQL command to compile or rows of data to insert into a database).

There is one output message buffer per connected cursor on the client computer. The database server maintains a buffer that is the size of the largest output message buffer on the client computer. The client builds a message in its buffer, then sends it across the network to the server's output message buffer. It is called an output message buffer because it is output from the client's perspective.

This keyword is optional. Configure this keyword in any client router section of SQL.INI, except for Windows NT.

Default value The default value is 1000 bytes. Despite the *outmessage* value, SQLBase dynamically allocates more space if necessary.

A large output message buffer does not necessarily improve performance because the buffer only needs to be large enough to hold the largest SQL command to compile or the largest row of data to insert. (Rows are always sent to the database and inserted individually except in bulk execute mode.) A large output message buffer can allocate space unnecessarily on both the client and the server, and it does not reduce network traffic unless bulk execute is on.

Example To set the output message buffer to 2000 bytes, specify:

```
outmessage=2000
```

partitions

Enables access to partitioned databases.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value By default, *partitions* is off (0) which means that:

- You can restore *main.dbs*.
- You cannot access partitioned databases. An exception is when you do not have a MAIN database. After creating the first dbarea, SQLBase creates *main.dbs* and sets partitions on (1).

Example To enable access to partitioned databases, specify:

```
partitions=1
```

password

Sets a server connection password for the server.

Specify the *password* keyword immediately after the *servername* keyword to force users to specify a password to perform administrative operations or to get server information.

A server connection password can be no longer than eight characters.

You can use a server connection password by itself if you are not using encrypted databases. If you are using encrypted databases, you can use a *server connection password* along with a *server security password* or you can set the password to an asterisk in SQL.INI and use only a server security password. For more, read *Server security password* on page 7-17.

This keyword is optional, but recommended for production databases. Configure this keyword in the server section of SQL.INI.

Default value This keyword has no default value.

Example To set server1's password to 'secret', specify:

```
servername=server1  
password=secret
```

preferrednameservice

Specifies a name service (DNS or Bindery) to use. A name service is something that converts a database name into an address and socket combination in order to establish a session with a database server. Setting this keyword reduces the time necessary for making a connection.

For details about configuring this keyword, read *Search order* on page 11-10.

The syntax is:

```
preferrednameservice=[NDS|BIN]
```

This keyword is optional. Configure this keyword in a client section of SQL.INI.

Default value For the NetWare 4.x, the default is NDS, followed by the Bindery.

If you do not set this keyword, SQLBase attempts to make a decision based on the following:

- Is directory services running?
If yes, use it to find the database.
- If no, try the NetWare bindery. Was the name found?
If yes, make the connection.

Example Here is an example of the *searchcontext* and *preferrednameservice* keywords:

```
[winclient.spxw4]
searchcontext=CN=ADMIN.O=Gupta
preferrednameservice=NDS
```

procwin

Determines the status of the Process Activity subwindow. The syntax for this window keyword is identical to that for the *mainwin* keyword, with the addition of a *closed* status. This indicates that the given subwindow was closed when you last used the **Save** option of the **File** menu.

The keyword is optional. This keyword is configured automatically in the server's GUI section (for example, [dbntsrv.gui]) when you use the **Save Settings** option in the server's **File** menu. Do not set a value for this keyword manually.

Example The syntax is:

```
procwin=status,x,y,cx,cy
```

The following values are valid for these settings:

- *Status*
Indicates the status of the window when you last used the **File, Save** option. Valid settings are: NORM for normal window status, MAX if the window is maximized, MIN if the window is minimized, or CLOSED if the window is closed.
- *x, y*
These values represent the position of the window. For Windows NT (4.0 and later), 2000, XP, and Server 2003, this is the window's top-left corner.

- ***cx,cy***

These values represent the size of the window. For Windows NT (4.0 and later), 2000, XP, and Server 2003, these are the coordinates of the window's lower-right corner. These values are system-dependent; do not manually modify or copy them from one system to another.

readonly

Allows users connecting to *any* of the databases on the server to use the read-only isolation level.

To enable read-only access to a *specific* database on the server, use SQLTalk's SET READONLY command or the *sqlset* API function with the SQLPISO parameter.

The read-only isolation level allows for a consistent view of data; for the duration of your session, the data in the database appears not to change. When readonly is enabled, SQLBase maintains a read-only history file that contains multiple copies of modified database pages. When you try to access a page which has been changed by another user, SQLBase retrieves a copy of the original page from the history file.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value By default, *readonly* is disabled (0) and users are prevented from going into read-only mode because read-only transactions can affect performance.

Example To enable read-only access, specify:

```
readonly=1
```

searchcontext

Specifies the part of the NDS tree that a client searches for the SQLBase server names and installed database names.

For details about configuring this keyword, read *Search context* on page 11-11.

If the client is non-NDS, the keyword is ignored.

Note: Be sure the *searchcontext* keyword is set to the same value as the *insertioncontext* keyword in the [server] section of SQL.INI.

This keyword is optional. Configure this keyword in any of these client sections of SQL.INI:

- SPX Windows client section
- TCP/IP Windows client section

Default value If no keyword is specified in the client section, the entire NDS directory tree is searched.

Example To specify that Windows NT (4.0 and later), 2000, XP, and Server 2003 SPX clients search a part of the Directory Tree identified as O=Gupta, specify:

```
[winclient.spxw4]  
searchcontext=O=Gupta
```

secureapi

You use this keyword to enable transmission security between SQLBase clients and servers to prevent unauthorized users from seeing data as it is transmitted across the communications medium.

Note: Depending upon the SQLBase edition you are using, you may be unable to use a higher security level. Attempting to change to a higher level than supported, will result in errors on the client.

For details about configuring this keyword read *Transmission security* on page 7-25. The table below shows the meaning of the settings of the SECUREAPI keyword:

Level	Value	Client meaning	Server meaning
None	0	Use no encryption	Accept any encryption level
Medium	1	Use medium encryption	Accept medium and high encryption levels only
High	2	Use high encryption	Accept high encryption level only

This keyword is optional. Configure this keyword in the client or server section of SQL.INI.

Default value For the client and server, this is 0 (none).

Example A client could have this in *SQL.INI*:

```
[win32client]
secureapi=2           ; always use high encryption
```

The server could have this in *SQL.INI*:

```
[dbntsrv]
secureapi=0           ; accept non-cripted messages
```

servername

Specifies a name for a multi-user server.

To perform administrative operations, a user must establish a connection to the server itself and specify the server's password (if one exists). This prevents unauthorized users from performing destructive operations. To establish (or break) a connection to a server, use SQLTalk's 'SET SERVER <servername>' command. SQLTalk's SHOW DATABASES command also accepts a server name.

The *sqlcsv* API function requires a server name as input and returns a handle which the *sqlds* functions requires as input. The *sqldb* API function requires a server name as input and returns a list of databases on a server.

Configuration of this keyword is recommended for those sites for which security is important. Configure this keyword in the server section of SQL.INI.

- Default value** If you don't configure the *servername* keyword, it defaults to *s###* where '###' is a random 3-digit number.
- The name can be no longer than eight alpha-numeric characters, it must begin with a letter, and it must be unique on the network.
- Example** To name a server 'server1', specify:
- ```
servername=server1
```
- You can optionally specify one or more communications library names after the server name. For example:
- ```
servername=server1,sqlws32
```
- You must also specify the communications libraries with the *comdll* keyword in the server's [*.*dll] section.

servernames

Specifies the names of one or more LAN Manager file servers to which a database server connects.

Configuration of this keyword is mandatory in a LAN Manager environment for remote connections; otherwise, it is unnecessary. Configure this keyword in the Named Pipes section of a SQLBase for Windows server's SQL.INI file ([dbntsrv.npipe]) or the [win32client.npipe] section of a Windows client's SQL.INI.

- Default value** This keyword has no default value.
- Example** To identify two file servers, specify:
- ```
servername=lanman1;lanman2
```
- Substitute appropriate LAN Manager server names for *lanman1* and *lanman2*.

## serverpath

Tells a client using the TCP/IP protocol how to locate SQLBase servers by associating a server name and one or more database names with a host's network name, IP address, or TCP port number. You also must specify at least one *serverpath* keyword in a server section if the server is connected via TCP/IP and is participating in a distributed transaction.

The syntax of the keyword is:

```
serverpath=servername,hostname[,tcp-port]/dbname[,dbname...][*]
```

The servername is the name of the server as specified in the server's SQL.INI file.

The hostname is the network name or the IP address of the machine on which the server is running.

The optional tcp-port is the TCP port number on which the host is listening for connection requests. You need to specify this only if the *listenport* keyword is specified in the TCP/IP section of the server's SQL.INI file.

The dbname is the name of a database on the server. You can specify more than one database name by separating each name with a comma. The asterisk (\*) is an optional wildcard symbol that you can use to match any database name.

This keyword is mandatory for clients accessing SQLBase servers through the TCP/IP protocol, or for servers participating in a distributed transaction through TCP/IP. Otherwise, it is unnecessary.

Configure this keyword in a TCP/IP communication section.

You can configure multiple *serverpath* keywords in the client's SQL.INI. This gives the client the flexibility to connect to databases on different servers from the same session. However, the following restrictions apply:

- Each *serverpath* must specify a unique host name or IP address.
- The database names specified in the *serverpath* statements must be unique in that section. For example, you cannot specify *demo* as a database name on two *serverpath* statements in the client's Windows Socket section.
- You cannot use more than one wildcard character (\*) in the section. That is, you cannot specify two *serverpath* statements, each with the wildcard (\*) in it.

## Examples

Here are two examples of serverpath statements:

This example shows that a SQLBase server called *server0* is installed on a host called *host1* and is listening on the databases *demo* and *demox* for connection requests.

```
serverpath=server0,host1/demo,demox
```

This example shows that a SQLBase server called *server1* is installed on a host with an IP address of 151.222.1.1, and is listening on the database account for connection requests.

```
serverpath=server1,151.222.1.1/account
```

The following example shows the use of multiple *serverpath* statements. In this example, if the client attempts to connect to database *ABC*, SQLBase matches *ABC* to the wildcard (\*) and then uses the second *serverpath* (and its IP address) to connect to it. However, if the client attempts to connect to *DEMO1*, the first *serverpath* is used.

```
serverpath=server1,198.206.11.10/demo1
serverpath=server2,198.206.11.20/demo2,*
```

## showmaindbname

Enables (1) or disables (0) the display of the MAIN database when you query the server for a list of database names.

This keyword is optional. Configure this keyword in any client router section of SQL.INI.

**Default value** By default, the display of the MAIN database is off (0).

**Example** To enable the display of the MAIN database, specify:

```
showmaindbname=1
```

## silentmode

Turns the display for a multi-user server on and off.

---

**Warning:** If you turn off the server display, there will be no visible window and you cannot shut down the server with Windows Task Manager. The only way you can shut down the server is through SQLTalk or the SQLBase API.

---

This keyword is optional. Configure this keyword in any server section of SQL.INI.

**Default value** By default, multi-user server displays are on (0).

**Example** To set the display of the server screens off, specify:

```
silentmode=1
```

## sortcache

Specifies the number of 1K (1024-byte) pages of memory to use for sorting. The value applies across the server. Each sort process is allocated its own set of pages equal to the sortcache setting. Sorting is done when you specify a DISTINCT, ORDER BY, GROUP BY, or CREATE INDEX clause, or when SQLBase creates a temporary table for join purposes.

Note that the combined values of *cache* and *sortcache* cannot exceed the memory resources of your system.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

**Default value** The default is 2000. (SQLBase accepts a maximum value of one million, but you should set this value based on how much memory you have in your system).

**Example**

To set the amount of memory to use for sorting to 1000, specify:

```
sortcache=1000
```

---

**Note:** Sortcache memory is not pre-allocated. The memory used is separate from the database cache, so ensure there is enough server memory for both.

---

**statwin**

Determines the status of the Server Status subwindow.

The syntax for this keyword is identical to that for the *mainwin* keyword, with the addition of a *closed* status. This indicates that the given subwindow was closed when you last used the **Save** option of the **File** menu.

This keyword is optional. This keyword is configured automatically in the server's GUI section (for example, [dbntrsv.gui]) when you use the **Save Settings** option in the server's **File** menu. Do not set a value for this keyword manually.

**Example**

The syntax of this keyword is:

```
statwin=status,x,y,cx,cy
```

**syswin**

Determines the status of the System Activity subwindow.

The syntax for this keyword is identical to that for the *mainwin* keyword, with the addition of a *closed* status. This indicates that the given subwindow was closed when you last used the **Save** option of the **File** menu.

This keyword is optional. This keyword is configured automatically in the server's GUI section (for example, [dbntrsv.gui]) when you use the **Save Settings** option in the server's **File** menu. Do not set a value for this keyword manually.

**Example**

The syntax of this keyword is:

```
syswin=status,x,y,cx,cy
```

**tempdir**

Specifies the directory where SQLBase places temporary files.

In the course of processing, SQLBase can create several kinds of temporary files: sort files, read-only history files, and general-use files.

Temporary files have names in the format: *sqlxxxx.tmp* where 'xxxx' is a randomly-generated number.

SQLBase uses asynchronous input/output to read and write temporary files.

You must set *tempdir* for read-only databases. Otherwise, configuration of this keyword is optional, though recommended for production sites to ensure that you do not accidentally run out of disk space.

Configure this keyword in any multi-user server section of SQL.INI.

**Default value** SQLBase places temporary files in one of three places in this order:

1. The directory specified by the *tempdir* keyword in SQL.INI.
2. The directory specified by the *tempdir* environment variable.
3. The current directory.

**Example** To specify d:\tmp as the temporary directory, specify:

```
tempdir=d:\tmp
```

The maximum length of the *tempdir* value is 260 characters. The maximum number of characters allowed in an *tempdir* directory path (excluding the filename which has a limit of eight characters and a three-character extension) is 246. This includes the drive letter, colon, leading backslash, and terminating null character.

Semicolons and commas cannot be part of the pathname. A directory name enclosed in single quotes can contain an embedded single quote which is indicated by TWO single quotes (for example: D:\'test'').

---

**Note:** Characters reserved in the Windows environment cannot be used in directory or file names; for example: <> : “ / \ |. Words reserved in the Windows environment cannot be used as file names; for example, *con* or *aux*.

---

## threadmode

Specifies whether to use native threads or SQLBase threads for NetWare or Windows NT (4.0 and later), 2000, XP, and Server 2003. For Windows 98/ME, SQLBase uses Windows 98/ME native threads only.

A value of 1 indicates SQLBase threads and a value of 2 indicates native threads for the current platform.

This keyword is optional. Configure this keyword in a SQLBase for NetWare 4.x server section or a SQLBase for Windows NT (4.0 and later), 2000, XP, and Server 2003 server section.

**Default value** By default, *threadmode* is 1, except on Windows 98/ME where the default is 2.

On NetWare platforms, if you are running in Ring 0, Gupta recommends using SQLBase threads which invoke stack switching. This should yield better

performance. Novell disallows stack switching in Ring 3, so be sure to set *threadmode* to 2 when in Ring 3.

**Example** To use native threads, specify:

```
threadmode=2
```

## threadstacksize

Specifies the stack size (in bytes) on NetWare 4.x.

This keyword is optional. Configure this keyword in the [dbnw\*] section of a SQLBase for NetWare 4.x server's SQL.INI.

**Default value** By default, *threadstacksize* is 10 kilobytes and the minimum value is 8192 bytes.

You should not decrease the default value. Running complex queries when *threadstacksize* is set to 8192 can result in a stack overflow error.

If you receive stack overflow errors, increase the value of *threadstacksize* by 512 bytes at a time.

**Example** To increase the stack size to 8704 bytes, specify:

```
threadstacksize=8704
```

## timeout

Specifies the time period (in minutes) that the server waits for a client to make a request. If the client does not make a request within the specified period, SQLBase rolls back the client session, processes, and transactions. The timeout clock restarts each time the client makes a request.

This keyword is transaction-specific, not cursor-specific.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

**Default value** The *timeout* value is 0 (infinite) by default, and the maximum value is 200 minutes.

**Example** To set a timeout period of one hour, specify:

```
timeout=60
```

## timestamps

Specifies the timestamp setting. Timestamps can be 0 (off) or 1 (on), indicating whether timestamps are displayed with each line of information output to the Process Activity window. Set a timestamp with either the Level menu on the GUI server, the SQLTalk SET TIMESTAMPS command, or the *sqlset* SQL/API call in conjunction with the SQLPTMS parameter.

This keyword is optional.

## timezone

Sets the value of SYSTIMEZONE, a SQLBase keyword that returns the time zone as an interval of Greenwich Mean Time. SYSTIMEZONE uses the expression (SYSTIME - TIMEZONE) to return the current time in Greenwich Mean Time.

Calculate the value of *timezone* as an interval in hours, where the interval is local time minus Greenwich Mean Time.

**Default value** By default, *timezone* is 0.

**Example** When it is 1:00pm in the Pacific Standard Timezone (PST), it is 9:00pm in Greenwich. This equates to 1 minus 9 which equals -8. Set:

```
timezone=-8
```

This keyword is optional. Configure this keyword in the server section of SQL.INI.

## users

Specifies the maximum number of client applications (processes) that can connect to a server simultaneously. This means, for example, that a server configured with *users*=5 could support five clients running one application each, or one client running five applications, or two clients — one running two applications and the other running three applications, and so on.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

**Default value** The default value of *users* is 128, and the maximum is 800.

**Example** To limit server access to 50 applications, specify:

```
users=50
```

## workalloc

Specifies the basic allocation unit of a work space. For example, if a SQL command requires 5000 bytes and the default value of 1000 is in effect, SQLBase makes 5 memory allocation requests to the operating system ( $5 * 1000 = 5000$ ).

If, on the other hand, you specify:

```
workalloc=2500
```

SQLBase makes only 2 memory allocation requests to the operating system ( $2 * 2500 = 5000$ ).

This keyword is optional. Configure this keyword in the server section of SQL.INI.

**Default value** The default is 1000 bytes.

# worklimit

Specifies a maximum memory limitation (in bytes) for SQL commands. SQLBase cannot execute any SQL command requiring more than 4000 bytes of memory.

This keyword is optional. Configure this keyword in the server section of SQL.INI.

Default value     The default is NULL, meaning that no memory limitation exists.

Example            To set the maximum memory limitation, specify:

```
worklimit=4000
```

# The Registry

On Windows NT/2000/XP/2003 and Windows 98/ME, SQLBase gets configuration information from the Windows registry. The SQLBase install program initially makes these entries in the registry.



*Windows registry hierarchy.*

SQLBase stores two types of configuration information in the registry:

- Configuration information that is used by SQLBase client applications.
- Service configuration information used only when running as a service (described in *Service information in the Windows Registry* on page 12-4).

The client configuration information is in:

```
HKEY_CURRENT_USER\Software\Gupta\SQLBase\servername
```

If you have multiple SQLBase servers installed there will be multiple registry sections, each with a different value of *servername*.

This section of the registry contains information about your SQLBase installation, including:

| Name                 | Data type | Example data                           | Description                                                                                                                               |
|----------------------|-----------|----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------|
| ConfigFileName       | String    | sql.ini                                | This documentation refers to the configuration file name as SQL.INI, but in your actual installation the name can be anything you choose. |
| ConfigFilesDirectory | String    | C:\Program Files\Gupta\SQLBase 850     | Location of configuration files such as SQL.INI, error.sql, message.sql, couuntry.sql, and main.ini.                                      |
| InstallDirectory     | String    | C:\Program Files\Gupta\SQLBase 850     | Location where SQLBase executable files were copied by installer. .                                                                       |
| ServerType           | String    | SQLBase Windows Server Unlimited Users | Type of server installed (Netware, 1-user, 5-user, etc.).                                                                                 |



## Chapter 4

# Databases

---

This chapter discusses various aspects of databases that a DBA needs to know:

- About databases
- Database cache
- Database files (transaction log files and temporary files)
- Partitioned databases
- Database auditing

## About databases

This section describes the structure, naming, and placement of databases in SQLBase.

### Database template (*start.dbs*)

SQLBase has a database template file called *start.dbs*. Each new database that you create is a copy of *start.dbs*.

This template file contains the system catalog tables that SQLBase maintains. It also reflects any changes to the database structure; an example is the changes to the structure between one version of SQLBase to the next.

If you are doing either of the following upgrades, you need to unload your old database files and then load them to the new SQLBase server:

- From a version prior to 8.5.
- From one platform to another (such as Windows 9x to Windows XP).

### Database names

SQLBase database names must conform to the following restrictions:

- They can be up to sixteen characters in length.
- They must begin with a letter and include only letters and numeric digits.
- They must have a *.dbs* file extension.

Unless you are manually copying the *start.dbs* file to a new file, do not specify a file extension; SQLBase automatically assigns each database a file extension of *.dbs*. For example, if you execute:

```
CREATE DATABASE test;
```

from a SQLTalk session, SQLBase creates a database called *test.dbs* in a subdirectory called \TEST. If the value of the *dbdir* keyword is C:\Gupta, SQLBase places *test.dbs* in the C:\Gupta\TEST subdirectory.

### dbname keyword

For all servers except Windows, you must configure a *dbname* keyword for each database on the server. Conversely, make sure that each *dbname* keyword in the server's configuration file (SQL.INI) has a corresponding database in the appropriate directory. If there is a *dbname* keyword whose associated database file does not exist, you receive error 401 ('Cannot open the database') when you try to connect to the database.

Issuing a CREATE DATABASE command automatically enters the database name in the SQL.INI file.

## Default database name, user name, and password

You must specify a database name, user name, and password when logging on to SQLBase. Client programs from Gupta like SQLTalk prompt you for these values when you attempt to connect to a database.

SQLBase determines the database name, user name, and password by checking the following entries in this order:

1. What you enter when prompted by a client program.
2. The *defaultdatabase*, *defaultuser*, and *defaultpassword* keywords in SQL.INI.
3. The default of DEMO, SYSADM, SYSADM.

Use the Gupta Connectivity Administrator if you are on Windows 98, ME, NT, 2000 , XP, or Server 2003, or your preferred text editor to change the *defaultdatabase*, *defaultuser*, and *defaultpassword* keywords in the section of SQL.INI for the local database server or router.

## Database size

When running on an operating system that has a file size limitation (such as Netware), SQLBase imposes a 2 gigabyte size limit for non-partitioned databases. If you try to extend a non-partitioned database beyond 2 gigabytes on such an operating system, SQLBase issues an error message.

---

**Note:** The 2 gigabyte size limit is imposed on all other files manipulated by SQLBase versions prior to 8.0, on any operating system. For example, you cannot create a partition (dbarea) greater than 2 gigabytes, or back up a partitioned database greater than 2 gigabytes to a single backup file. This limitation does not apply for version 8.0 and higher on Win32 operating systems.

---

To support backups of databases greater than 2 gigabytes, SQLBase lets you perform backups to multiple segments. Read *Chapter 8, Backing Up and Restoring Databases* for more information.

## SIZE parameter

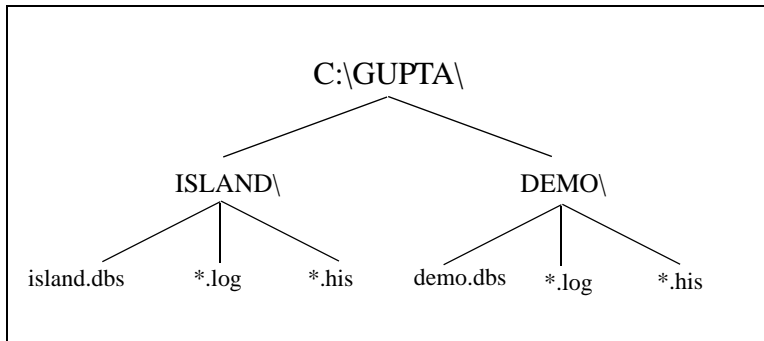
SQLBase uses the SIZE parameter for specifying the size of databases or files in specific operations. Depending on the operation, size can be specified in kilobytes or megabytes. For further details, read the following topics in the *SQLBase SQL Language Reference* manual:

- CREATE DBAREA
- ALTER DBAREA

- Control filename
- Audit filename

## Database location

SQLBase requires the existence of at least one ‘home’ database directory. By default, this is C:\Gupta. For each non-partitioned database you create on the server, SQLBase creates a subdirectory within the home database directory. The name of the subdirectory is the same as the database (minus the file extension). The subdirectory contains a database (.dbs) file, transaction log files (\*.log), and may contain read-only history files (\*.his) for a database.



*Default database directory organization*

### The dbdir environment variable and keyword

You can place databases on different drives and in different home database directories by configuring either the *dbdir* environment variable or the *dbdir* configuration keyword.

In addition to editing the configuration file (SQL.INI) directly, both SQLTalk and the SQL/API let you change the value of the *dbdir* keyword. Use SQLTalk’s SET DBDIR command or the SQL/API’s *sqlset* function with the SQLPDBD parameter to do so.

You can specify a path with one or more directories. The maximum length of the *dbdir* value is 255 characters. The format is the same as for the PATH command:

```
dbdir=d:\Gupta;d:\devel;d:\test
```

For SQLBase Server for NetWare, be sure to include the volume (for example, db:):

```
dbdir=db:\Gupta
```

You should specify a home database directory with a drive letter and path name. If you omit the drive letter, the current drive is assumed.

SQLBase looks for a database in these places in this order:

1. The directory pointed to by the *dbdir* configuration keyword or by the *dbdir* environment variable.
2. The current directory.
3. The \SQLBASE directory on the current drive.
4. The root directory on the current drive.

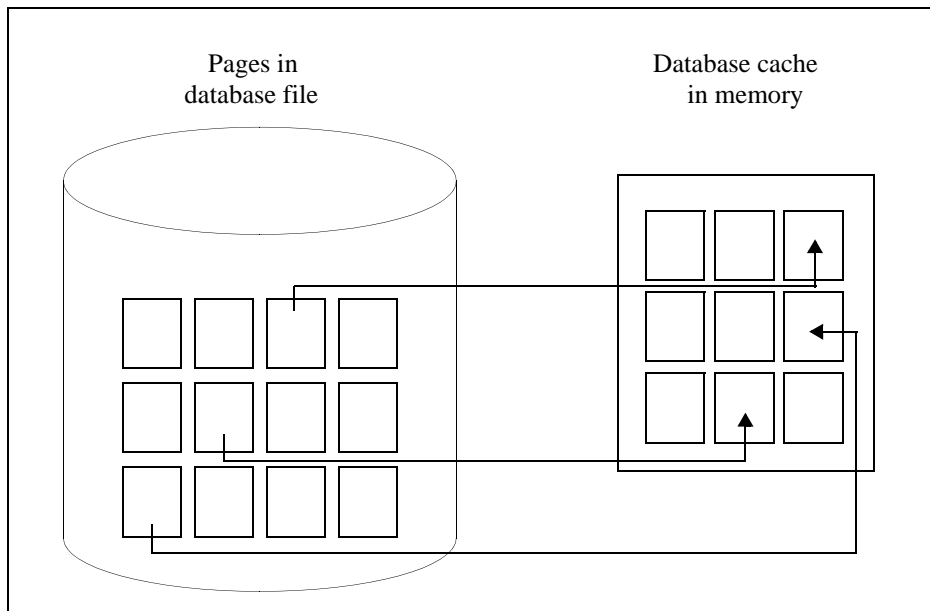
SQLBase creates a new database in the first directory on the *dbdir* path or in the current directory if *dbdir* is not specified.

If the configuration file contains two *dbdir* keywords, the second setting overrides the first.

For partitioned databases, the *dbdir* keyword specifies the directory that contains the MAIN database. In turn, the MAIN database holds the information about where partitioned databases and their log files are stored.

## Database cache

SQLBase uses a cache to optimize database input and output. The cache is an area of computer memory on the database server machine that contains copies of pages that users are reading or writing.



*Database pages in cache*

## Pages

A page is a unit of data from the database file that contains one or more rows of table or index data. It is the basic unit of physical disk storage in a database. Pages are stored as linked lists.

A page is 1024 bytes (1 kilobyte) in size.

When a user reads or writes a row or index, SQLBase checks to see if the page in which it is located is already in cache. If the page is not in cache, SQLBase copies it to the cache. If the page is already in cache, SQLBase uses the copy in cache. This reduces disk input and output. It is much faster to read or write to computer memory than to read or write to a disk file.

## Committing

When an implicit or explicit commit happens, SQLBase writes a commit record to the transaction log on disk. However, the database page in cache is written back (flushed) to the *.dbs* file based on an LRU (least-recently used) algorithm. The information in the transaction log is enough to completely recreate the update to the database if there is a crash, so there is no need to flush the page from cache immediately after a commit.

## Checkpoint time interval

A fuzzy checkpointing scheme minimizes the time to perform crash recovery. *Fuzzy checkpointing* means that modified database cache pages are marked at one checkpoint and then written at the next if they have not already been written by normal cache management. A transaction log record is written which describes the active transactions at each checkpoint. Logs containing the last two checkpoint log records are pinned to the disk (not deletable), as these checkpoints are used to quickly establish a redo start point for the rollforward phase of crash recovery.

The checkpoint time interval parameter governs how often a recovery checkpoint operation is done. You can set the checkpoint time interval with SQLTalk's SET CHECKPOINT command or the SQL/API's *sqlset* function. The default is every minute.

Depending on the applications running against the server, a checkpoint operation may affect performance. In this case, you can increase the checkpoint interval until the desired performance is gained.

## Database files

In addition to an open database file, SQLBase may need to open the following files as well:

- Transaction log files
- Temporary files
- Error file

Remote servers may also have these files open:

- Process Activity log file
- Netlog file

## Transaction log files

A transaction log file contains before- and after-images of changes to the database as well as log records for transaction control (checkpoints, for example). Transaction control includes documenting when a transaction started and how it ended. COMMIT and ROLLBACK are examples of the latter. A transaction log has three functions:

- Rollback

Transaction log files contain the data needed to rollback a transaction. A rollback can be initiated by either SQLBase or the user.

- Crash recovery

Transaction log files contain data necessary to bring a database back to a consistent state after a crash which can occur as the result of a power failure or operator error. A database is in an inconsistent or crashed state if it was not shut down gracefully; an example is when you reboot or power off the server computer without first exiting SQLBase using the Esc key after all sessions have exited. SQLBase performs crash recovery automatically whenever a user connects to a crashed database that has just been brought back online.

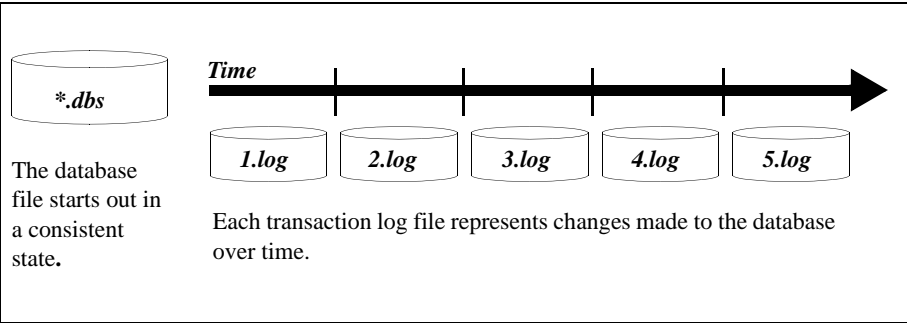
- Media recovery

Transaction logs, when saved in backup form along with a database, contain data needed to restore that database if damaged by a media failure. A media failure can occur as the result of a crashed hard drive, or a number of other hardware problems. In a situation like this, you can only achieve full recovery through the static restore of an operating system backup.

Transaction logs help ensure data consistency. If a transaction is rolled back or if a system or media failure occurs, SQLBase uses the transaction logs to restore the database to its original state.

For the reasons listed previously, transaction log files, like database files, should be backed up regularly so that they can be restored if necessary.

**Note:** Never delete transaction log files. SQLBase automatically deletes log files either when they are backed up, or when they are no longer needed for transaction rollback or crash recovery, depending on whether LOGBACKUP is on or off. A database (.dbs) file is useless without its associated log files.



SQLBase creates a new log file upon the first connect to a database. As changes occur to the database, SQLBase writes log records documenting those changes in the active log file.

SQLBase creates a new log file when the current log file reaches its specified maximum size (see *Log file size* on the next page).

The first log file is named `1.log`. Subsequent log files are named sequentially, so `2.log` is created after `1.log`, `3.log` is created next, and so on. The highest log file number is 99999999. Internally, log files contain timestamps and other values used to ensure that SQLBase can identify them in the correct order. Nevertheless, a validity check is done to verify that the log file numbers are correct, so log files should never be renamed or renumbered.

SQLBase uses the Write-Ahead Log (WAL) protocol that ensures that the log records for a modified page are written before the modified page (in database cache) is written back to the disk. This ensures that SQLBase can undo any uncommitted changes that have been written to disk if there is a crash.

### Log file directory

SQLBase creates log files for non-partitioned databases in the home database directory by default. To improve performance, you can redirect the log files to a different directory using the `logdir` keyword in `SQL.INI`. Within this directory, subdirectories are created for each database, and it is in these subdirectories that the database log files are placed by SQLBase.

Partitioned databases do not use the *logdir* keyword. The MAIN database has information about where a partitioned database's log files are stored.

If you redirect log files to another drive, that drive should be on the database machine and should not be on a network file server. This recommendation is based on the reasoning that you will still gain fault tolerance, an increased capacity for log files, and a reduced chance of losing log files if the drive that the database is on fails. The advantage of redirecting log files to a database machine drive is that performance will not suffer the overhead incurred by increased network traffic that results when log files are redirected to a network file server drive.

## Log file size

The maximum log file size is set with the SQLTalk SET LOGFILESIZE command or the *sqlset* API function. When the log file grows past this size, SQLBase closes it and creates a new log file. The default size of the log file is 1 megabyte. The smallest size is 100,000 bytes.

A large log file will improve database performance slightly because log files will need to be created less often, but if the log file is too large, it wastes disk space.

## Log file growth (incremental or preallocated)

By default, the current log file grows in increments of 10 percent of its current size. This uses space conservatively but can lead to a fragmented log file which may adversely affect performance.

By enabling log file *preallocation*, log files are created full size and do not grow incrementally. You enable preallocation with the *logfileprealloc* keyword in SQL.INI, or with the SQLTalk SET LOGFILEPREALLOC command, or the API function *sqlset*.

## Log file space availability

For the Windows 32-bit environments and NetWare 3.x or higher, SQLBase queries the disk space available *before* opening a new log file when the current log file for a non-partitioned database has reached its specified maximum size. If the space available is inadequate to ensure database integrity should a system failure occur, SQLBase selectively rolls back any transaction that is causing excessive logging. Transactions that are pinning the earliest active log file are rolled back and the following error message is displayed:

```
03926 LOG LSC Log disk space critically short
```

When this message is displayed, check the log disk space immediately. SQLBase must unpin the log files to free up disk space and to do this, must roll back the transaction that is currently pinning the log files.

When SQLBase rolls back the transaction, it requires free disk space to process:

- Information from logs files the transaction has pinned that must be released to preserve the integrity of the database.
- Generation of new log files to reflect the rollback.

If unpinning the log files associated with the transaction fails to recover log disk space, SQLBase is forced to shut down.

---

**Note:** Although you may find you still have disk space, it may not be adequate to fulfill SQLBase recovery requirements. The space required for recovery is twice the size of the data required for processing. SQLBase also requires space allowance for a second recovery attempt, should the first fail.

---

If you experience unwarranted transaction rollbacks, you can disable SQLBase from checking for available log file space. For details, read the description for the *disablelogspacecheck* keyword in *Chapter 3, Configuration: SQL.INI and the Registry*.

SQLBase checking for available disk space does not apply to partitioned databases. This is because partitioned databases use “private” logging disk space. In a partitioned database, you can extend the logging space which can span multiple drives.

## Transaction span limit

Long running transactions pin log files to disk that could otherwise be released (and deleted if LOGBACKUP is off). You can limit the number of logs pinned down by active transactions by specifying the transaction span limit with the SQLTalk SET SPANLIMIT command or the API function *sqlset*. If you set a limit, SQLBase will roll back long running transactions that exceed the limit, thus allowing pinned log files to be released.

---

**Note:** It is important that you understand the transactions at your site before enabling this feature.

---

You specify the transaction span limit as the number of log files that an active transaction can span. As new log files are created, SQLBase checks this limit for all active transactions. Any transaction that violates the limit is rolled back.

By default, transaction span limit checking is disabled.

## Pinned log files

Log files are pinned (held on disk) for several reasons:

- The log file is currently active.
- The log file(s) is needed to potentially rollback one or more active transactions.

For example, if the earliest active transaction was first logged in log 11 and processing is currently at log 16, logs 11 through 16 are pinned.

- The log file is needed for crash recovery if a failure occurs.

This is partly covered by the preceding requirement, since crash recovery would need to rollback any transactions active at the time of the crash. In addition, logs containing the last two checkpoint log records are pinned, as these checkpoints are used to quickly establish a redo start point for the rollforward phase of crash recovery. Checkpoints are generated every minute, by default, meaning that at least two minutes worth of log records will be pinned on disk.

- LOGBACKUP has been enabled and inactive log files have not been backed up.

Log files become candidates for deletion under the following conditions:

- When the current log file becomes full and is rolled over or you force a log rollover. Note that a forced log rollover also occurs when you backup offline, and reset the NEXTLOG parameter by either a SQLTalk SET NEXTLOG command or a call to the *sqlset* API function, specifying the SQLPNLB parameter.
- When SQLBase unpins a log file by forcing a rollback on a transaction that has pinned the log file.
- When log files are backed up.
- If LOGBACKUP is off and the log is no longer needed for rolling back transactions or crash recovery.
- When database logging is turned off by a SQLTalk SET RECOVERY OFF command.

An attempt to delete unpinned log files is *not* made at every commit/rollback, as this would affect performance.

## Log file deletion

If media recovery is important to your site, you will need to set the LOGBACKUP parameter on to specify that logs are to be backed up before they are deleted. Use the SQLTalk SET LOGBACKUP command or the *sqlset* API function.

The DBA is responsible for carrying out routine backup procedures at which time SQLBase automatically deletes log files. By default, LOGBACKUP is off, so that SQLBase deletes log files as soon as they are not needed to perform transaction rollback or crash recovery. This is done so that log files do not accumulate and fill up the disk.

If log files are backed up off-line, the NEXTLOG parameter needs to be set once the database is brought back online. This parameter alerts SQLBase to the fact that an off-line backup has occurred and that there are logs eligible for deletion.

## Releasing a log file for backup or deletion

The SQLTalk RELEASE LOG command and the API function *sqlrel* force the release of the currently active log file, making it available for backup or deletion by SQLBase.

SQLBase closes the current log file and creates a new one automatically when the current log file becomes full (this is called log *rollover*). The RELEASE LOG command and the *sqlrel* function let you back up a log file without waiting until it becomes full. This allows you to get the most current backup possible.

The RELEASE LOG command is not needed if the BACKUP SNAPSHOT command was used to make the backup. BACKUP SNAPSHOT automatically forces a log rollover.

If LOGBACKUP is off and you do a RELEASE LOG or *sqlrel*, the log file will not be deleted until the *next* log rollover.

## Turning recovery off

If a system failure occurs while recovery is off, your database will be in an inconsistent state and you can lose referential integrity.

You should only turn off recovery/logging if you can afford to lose your data in the event of a system crash or media failure. Use SQLTalk's SET RECOVERY OFF command (or connect using *sqlcnr*) to do this.

Once you set recovery off, the settings for autocommit, bulk execute, cursor-context preservation, restriction mode, rollback, scroll mode, isolation level, loadversion, and timeout will revert back to their defaults. Be aware that with recovery off, rollback and autocommit are meaningless.

You should back up the database and log files before you turn recovery off.

## Temporary files

In the course of processing, SQLBase single-user servers and multi-user servers can create several kinds of temporary files:

- Sort files
- Read-only history files
- General-use files

SQLBase names temporary files *sqlxxx.tmp* where *xxx* is a randomly-generated serial number. It places temporary files in one of three places, in this order:

1. In the directory specified by the *tempdir* configuration keyword
2. In the directory specified by the *tempdir* environment variable
3. In the current directory

### Sort files

Sort files contain the end result of sorts specified by a **DISTINCT**, **ORDER BY**, **GROUP BY**, or **CREATE INDEX** clause. SQLBase creates one sort file per sort clause.

A sort file is separate from a general-use file even if the data being sorted is part of the result set data contained in the general-use file. Once the result set is built in the general-use file, SQLBase deletes the sort file.

### Read-only history files

Read-only history files contain multiple copies (before-images) of changed data pages. History files enable SQLBase to offer read-only transactions a consistent view of the data as it existed when the read-only transaction began. SQLBase creates one read-only history file per database.

**Read-only transactions.** Read-only transactions may affect performance, so they are disabled by default.

To override the default setting and enable read-only mode for all databases on a server, set the *readonly* configuration keyword value to 1. To override the default setting and enable read-only mode for a specific database, use SQLTalk's **SET READONLY** command or the SQL/API's *sqlset* function with the **SQLPROD** parameter.

**History file size.** If you have enabled read-only mode, you can specify the size of the history file in kilobytes with SQLTalk's **SET HISFILESIZE** command or the SQL/API *sqlset* function. The default history file size is 1000 kilobytes (1 megabyte).

**Read Only isolation level.** When read-only mode is on, you can use the Read Only (RO) isolation level. This isolation level provides a before-image view of the data as it was when the SELECT command was executed.

The Read Only isolation level places no locks on the database and you can only use it for reading data.

## General-use files

General-use files contain result sets, temporary tables, temporary indexes used in processing joins, and hash join information. SQLBase creates one general-use file per database.

## How SQLBase uses read-only history and general-use files

Historically, SQLBase created temporary files in pairs and used them in a cyclical manner. When the first file reached 1000 pages in capacity, SQLBase began writing to the second file. This was known as a rollover. When SQLBase determined that none of the transactions begun in the first file were needed any longer, it unpinned and deleted the file and created another in its place. Ideally, this happened before the second file reached 1000 pages in capacity. If not, the second file continued to grow.

Currently, SQLBase does not create temporary files in pairs. Instead, it creates one temporary file and recycles the space in that file. For example, if you are in result set mode, committing a transaction or compiling or executing another SQL statement in the same transaction frees the result set created by your *last* SQL statement. This allows SQLBase to release the space in the temporary file occupied by that result set, and enables another transaction to claim that space for its own processing. By recycling already-existing space in this way, SQLBase can prevent temporary files from growing too large.

In addition, SQLBase attempts to delete the temporary file whenever possible. If none of the transactions in the temporary file are needed any longer, SQLBase deletes it. For example, assume that you are the only user connected to a database and you are in result set mode. If you compile and execute a query, SQLBase builds a result set in the temporary file. If you then re-compile and re-execute another query, SQLBase frees the space allocated in the temporary file for your first result set and deletes the temporary file. The results of your second query will be put into a *new* temporary file.

If you are in result set mode and you set cursor-context preservation on for one of a transaction's cursors, a commit (unlike a re-compile or re-execute command) does *not* free the result set. The space in the temporary file occupied by the result set remains claimed, preventing other transactions from using it.

If you are in restriction mode, the result sets created by each re-compile and re-execute are not freed. Neither are they freed when you issue a commit. That is

because these result sets are needed to serve as the data sources for each successive query you issue.

Ultimately, if all existing space in the temporary file is claimed, SQLBase must increase the size of the temporary file to enable other transactions access to it.

## Partitions

As databases grow, disk space becomes critical. SQLBase enables you to spread your databases across multiple disk partitions to handle larger databases and increase performance. SQLBase provides a way for you to use partitions on all available platforms. In this way, database size is only limited by the total amount of disk space available on the server.

SQLBase enables you to define multiple named *database areas*, which are physical files or devices where database data and log files can be stored. These areas can be spread across multiple disk volumes to take advantage of parallel disk input/output operations.

You can spread database log files across different devices or disks. This allows concurrent log backups.

## What is a partition?

Partitioning maximizes disk space by dividing a disk into different database areas on multiple disks. Partitions can be on different disks on the same system.

---

**Important:** Partitions must be on a local drive on the computer where SQLBase runs; they cannot be accessed over a network.

---

A database can extend across several partitions without slowing down performance. Performance may actually be increased by allowing simultaneous writes to the database, as well as by reducing the time to write to the database and log files.

Partitioning can be turned on and off with the following SQLTalk commands:

```
SET PARTITIONS ON;
```

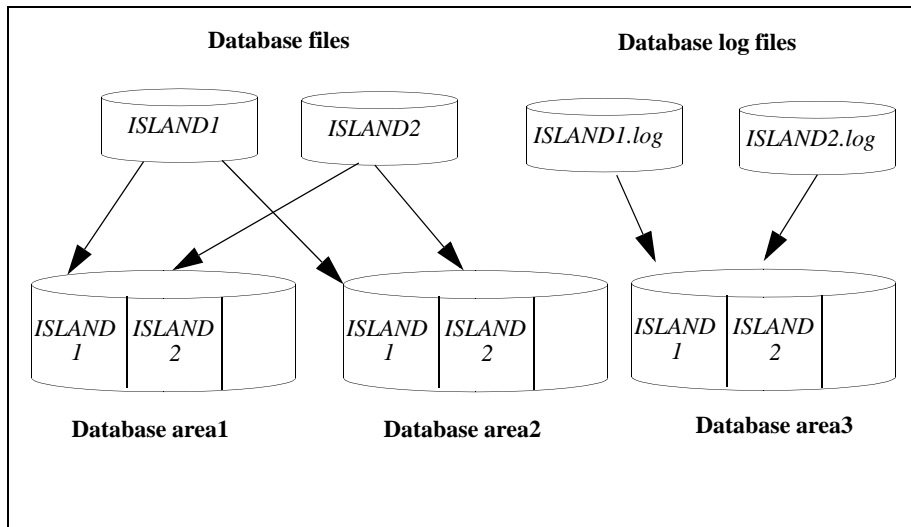
```
SET PARTITIONS OFF;
```

You must disable partitioned databases before you restore the MAIN database from backup, and you must re-enable them once recovery is complete.

## What are database areas?

A database area is a portion of a disk specified by the user for the storage of database and/or log files. A database area can reside either on a normal file system or a NetWare file server volume. You can specify many database areas of various sizes and give them names.

Disk space is distributed from a database area to a particular database in units called extents. An extension unit is the amount of contiguous disk space in a database area which is allocated to a database or log file when that file is extended. The default extension unit value is 1 megabyte but this can be set to higher multiples of 1 megabyte.



*Databases and log files spread across three database areas*

As you can see in the above figure, there are two sample ISLAND databases and log files spread across three database areas. You can accomplish this by creating the database areas, assigning them to storage groups, and then creating databases in the storage groups.

You can increase or decrease the allocation size from the default of 1 megabyte, by using either:

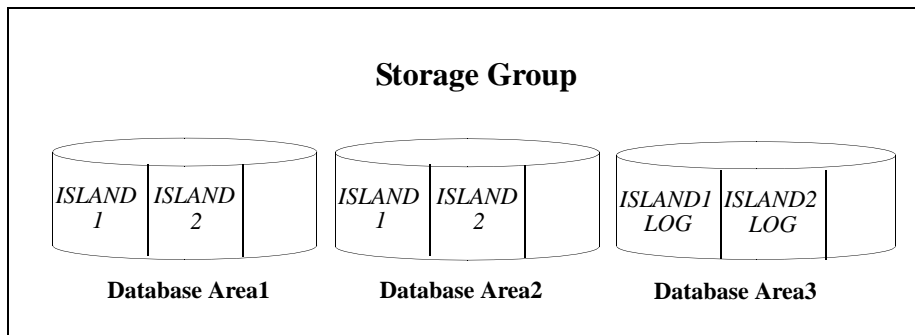
- SQLTalk's SET EXTENSION command.
- The SQL/API's *sqlset* function with the SQLPEXS parameter.

## What are storage groups?

A storage group is a named list of database areas which is used to organize and specify partitioned databases. A similar concept is an electronic mail group, which is a named list of electronic mail recipients.

You can declare storage groups for many different functions. For example, you could declare a storage group for high-access databases that lists the areas from one or more fast disks. Another storage group could list areas for low-access databases which reside on slower disks. A database area can appear on multiple storage group lists.

Using storage groups allows you to store databases on different volumes. Storage groups also allow the database system to automatically handle most of the details of allocating space. In the following graphic, the three database areas form one named storage group.



*Three database areas form a storage group*

When you create a database, you can specify a storage group to designate where the file space for that database is located. You can also specify a default storage group to handle databases that do not explicitly designate a storage group. If you do not specify a default storage group, the database is created in the normal file system.

Each database area within a storage group may contain extents allocated to different databases. In addition to selecting a storage group for the database, you can specify a storage group for the logs of that database. This enables you to place the logs on a disk separate from the database for better performance and data integrity.

## The MAIN database

SQLBase creates a MAIN database under either of the following conditions:

- If you create a partitioned database.
- If you enable commit service for a database by setting the *commitserver* keyword in the SQL.INI file. Read *Chapter 9, Distributed Transactions* for information on commit service.

The MAIN database stores control information that you can query:

- Partitioned database file names
- Database area and extent names and sizes
- Storage group names
- Participants in a distributed transaction
- Two-phase commit operations

To connect to the MAIN database, specify the name of the server on which the MAIN database resides when you are prompted to enter the database name.

You should backup the MAIN database whenever you backup partitioned files. You can restore the MAIN database, but you cannot change its columns and rows.

You cannot create a database called MAIN yourself, nor should you have an existing database called MAIN. In addition, do not delete the MAIN database.

Read *Appendix A, System Catalog Tables* for a list of tables and views in the MAIN database.

### MAIN's initialization file (main.ini)

The *main.ini* file is an initialization file used only for creating the MAIN database.

The *main.ini* file must be in the same directory as that from which you start the database server. This means that *main.ini* must be in the your current directory when starting in single-user mode and in the server software directory when in multi-user mode. In both cases, C:\Program Files\Gupta is the default.

The *main.ini* file creates the GUEST/GUEST account, as well as creates views, synonyms, and stored commands which enable you to access the partitioned database information contained in the MAIN database.

For example, to find out if a database is partitioned, connect to the server name and use the guest account to log on to MAIN. Then query the public synonym DATABASES. As another example, to get information about available space, you can query the public synonym FREEEXTS.

Read *Appendix A, System Catalog Tables* for a list stored commands contained in the *main.ini* file.

## Database auditing

SQLBase's database auditing feature allows you to audit database activities and performance. For example, you can monitor who logs on to a database, which tables they access, and if they attempt to access data for which they do not have privileges. You can also measure server performance by gathering information on command

execution time and long-running transactions. To gather this data, you start one or more *audit operations* that write output to an associated audit file.

You can have up to 32 active audit operations running concurrently. However, generally you do not need more than one or two, since you can record different types of information within each audit. Also, be aware that each additional audit operation can affect performance.

All concurrent active audits must have unique names.

## Audit file

An audit operation writes its output to an *audit file*. This is a simple ASCII file that you can print or access through either an editor or SQLConsole.

The audit file collects various types of system and performance information depending on the audit type and category you choose. You can also direct a message to an audit file to document a system activity.

An audit file name has the format *<auditname>.x*, where *x* is an ascending value. For example, starting an audit operation with the name *myaudit* generates an output file called *myaudit.1*.

You can record information to one audit file of unlimited size, or to a series of sequential files that have a specified size. When a file reaches this defined size, SQLBase automatically generates a new file. For example, if you specify a maximum size of 100 kilobytes when you start the *myaudit* audit, SQLBase automatically creates and starts writing output to *myaudit.2* when *myaudit.1* reaches a size of 100 kilobytes.

Since the sequence of audit files is controlled by the file extension, you can theoretically have up to 1000 audit files for each individual audit (though this may not be practical for real-time use). The audit file extension sequence ranges from *<auditname>.1* to *<auditname>.999*. After *<auditname>.999*, the next file is called *<auditname>.0*, and then wraps around to begin the sequence again with *<auditname>.1*.

You can automatically delete old audit files by telling SQLBase to keep only a certain number of files besides the current file. For example, if you specify a KEEP value of 2 for an audit called *myaudit*, (with the SQL START AUDIT command), SQLBase automatically deletes *myaudit.1* when it creates *myaudit.4*, since it can only keep two old files (*myaudit.2* and *myaudit.3*).

## Starting and stopping an audit

To start an audit, use the `START AUDIT` command. This command creates the following entry in the appropriate section of your configuration file (`SQL.INI`):

```
AUDIT=[type],auditname,[directoryname],[size-integer],
[append-integer],[keep-integer],[OVERWRITE],x,...
```

For example:

```
AUDIT=GLOBAL,SECURITY,C:\Gupta,1000,1,1,OVERWRITE,1,2,3
```

For descriptions of these parameters, read the `START AUDIT` command documentation in the *SQLBase SQL Language Reference*. The `x` field represents an information category, which is described in the next section.

An audit remains active while the server is running. It stops when you shut down the SQLBase server, but restarts when you bring the server back up. To stop an audit operation, use the `SQL STOP AUDIT` command.

## Types of audit operations

SQLBase supports two types of audit operations: *global* and *performance*. These two audit types record different types of information. You specify what type of information to record by indicating its category when you run `START AUDIT`.

Both audit types also automatically record the following general information in the audit file:

| Category | Description                                                                                                      |
|----------|------------------------------------------------------------------------------------------------------------------|
| 0        | Start of audit.                                                                                                  |
| 998      | Records when a message was issued to an audit file. Read <i>Sending a message to an audit file</i> on page 4-25. |
| 999      | End of audit.                                                                                                    |

The next two subsections describe the global and performance audit categories, and what type of information each category records. For examples and detailed information about each category, read the following sections.

### Global

Use this type of audit to monitor activities for the entire SQLBase system. This audit type provides global information for the server.

The following table shows the types of information you can monitor with a global audit, and their category numbers. Categories with an asterisk (\*) contain important security information that you may want to record on a continuing basis.

| Category  | Data collected                                           | Description                                                                                                                                       |
|-----------|----------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------|
| <b>*1</b> | Rejected logons                                          | Records unsuccessful login attempts. Useful to see if a user tried to access a restricted database.                                               |
| <b>*2</b> | Security violations                                      | Tells you if a user tried to access data without proper privileges.                                                                               |
| <b>*3</b> | Valid logins/logoffs                                     | Records all valid logons, telling you when a user first connected, and whether he/she disconnected. Use it to find out what users were logged on. |
| <b>4</b>  | Valid connects/disconnects                               | Records CONNECT and DISCONNECT statements.                                                                                                        |
| <b>5</b>  | Database creates, drops, installs, and deinstalls        | Records CREATE DATABASE, DROP DATABASE, INSTALL DATABASE, and DEINSTALL DATABASE commands.                                                        |
| <b>6</b>  | Recovery operations                                      | Records all ROLLBACK commands.                                                                                                                    |
| <b>7</b>  | Backup and restore operations                            | Records all BACKUP and RESTORE commands.                                                                                                          |
| <b>8</b>  | Database Lock Manager deadlocks and timeouts             | Records information about deadlocks and timeouts.                                                                                                 |
| <b>9</b>  | Table access information (queries)                       | Tells you which users accessed which tables.                                                                                                      |
| <b>10</b> | Table update information (inserts, updates, and deletes) | Records database manipulation language commands (DML), and which users issued the commands.                                                       |

## Performance

Use this type of audit to monitor performance for a particular operation. A performance audit records how long a SQLBase operation takes, and is useful for performance analysis and tuning.

A performance audit records the performance of the following operations:

| Category | Data collected                                              | Description                                                                                                           |
|----------|-------------------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------|
| 101      | Connects and Disconnects                                    | Tells you how long it takes users to connect to and disconnect from a database.                                       |
| 102      | SQL command compilation, execution, storage, and retrieval. | Tells you how much time SQLBase takes to compile, store, retrieve, and execute a particular SQL statement.            |
| 103      | End of transaction.                                         | Tells you when a transaction ended, and how long the transaction took. Useful for locating long-running transactions. |

Record formats

This section provides the syntax of the different audit categories as they appear in the audit file. It also provides examples of the general record layouts: START AUDIT (category 0) and STOP AUDIT (category 999).

Many of the record layouts contain one or more of the following general fields:

| Field Names        | Description                                                                                                          | Format                                                                                                                                                                                                                                                            |
|--------------------|----------------------------------------------------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <i>datetime</i>    | The date and time the information for this record was recorded.                                                      | <b><i>dd-mon-yy hh:mi:ss.99</i></b><br><br>Read the <i>SQLBase SQL Language Reference</i> for information on this date/time format.                                                                                                                               |
| <i>database</i>    | The name of the database                                                                                             | Database name, without the <i>.dbs</i> extension. For example:<br><br>ISLAND                                                                                                                                                                                      |
| <i>clientname</i>  | The name that is displayed in the CLIENT NODE field on the SERVER STATUS display.                                    | Client name; for example:<br><br>WinUser                                                                                                                                                                                                                          |
| <i>elapsedtime</i> | This field appears in the performance audit file. It shows the amount of time SQLBase took to complete the activity. | Depending on the length of time the activity takes, this field can appear in one of three formats:<br><br><i>ss.99</i><br><i>mi:ss.99</i><br><i>hh:mi:ss.99</i><br><br>Read the <i>SQLBase SQL Language Reference</i> for information on these date/time formats. |

## START AUDIT and STOP AUDIT (categories 0 and 999)

This section shows examples of the START AUDIT and STOP AUDIT records written to the audit file.

| Record                                         | Format                                                                                                                      |
|------------------------------------------------|-----------------------------------------------------------------------------------------------------------------------------|
| Start global audit record<br>(category 0)      | 000,datetime,GLOBAL,auditname,x,x,x...<br>where <i>x</i> represents an audit category specified in the START AUDIT command. |
| Start performance audit record<br>(category 0) | 000,datetime,PERFM,auditname,x,...<br>where <i>x</i> is an audit category specified in the START AUDIT command.             |
| Stop audit record (category 999)               | 999,datetime,auditname                                                                                                      |

## Global audit categories

This section shows examples of all global audit categories.

| Record                                                              | Format                                                                                                                          |
|---------------------------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------------|
| Rejected logon<br>(category 1)                                      | 001,datetime,error#,database,username,clientname                                                                                |
| SQL security violation<br>(category 2)                              | 002,datetime,error#,database,username,clientname,tablename                                                                      |
| Valid logon/logoff<br>(category 3)                                  | 003,datetime,LOGON,database,username,clientname<br>003,datetime,LOGOFF,database,username,clientname                             |
| Valid CONNECTS/<br>DISCONNECTS<br>(category 4)                      | 004,datetime,CONNECT,database,username,clientname<br>004,datetime,DISCONNECT,database,username,clientname                       |
| CREATE, DROP,<br>INSTALL, and<br>DEINSTALL<br>DATABASE (category 5) | 005,datetime,CREATE database,<br>005,datetime,DROP database<br>005,datetime,INSTALL database<br>005,datetime,DEINSTALL database |
| Recovery operation<br>(category 6)                                  | 006,datetime,database,username,clientname                                                                                       |

| Record                                                    | Format                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               |
|-----------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| BACKUP and RESTORE operations (category 7)                | 007,datetime,BACKUP DATABASE,database,username,clientname<br>007,datetime,BACKUP LOGS,database,username,clientname<br>007,datetime,BACKUP SNAPSHOT,database,username,clientname<br>007,datetime,RELEASE LOG,database,username,clientname<br>007,datetime,RESTORE DATABASE,database,username,clientname<br>007,datetime,RESTORE SNAPSHOT,database,username,clientname<br>007,datetime,ROLLFORWARD CONTINUE,database,username, clientname<br>007,datetime,ROLLFORWARD END,database,username,clientname<br>007,datetime,ROLLFORWARDSTART,database,username,clientname                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Database lock manager deadlocks and timeouts (category 8) | 008,datetime,DEADLOCK,database,user,clientname,xx,nnn,group#,<br>page#,lock type, SQL STATEMENT...<br><br>008,datetime,TIMEOUT,database,user,clientname,xx,nnn,group#,<br>page#,lock type, SQL STATEMENT...<br><br>008,datetime,LOCK HOLDER,database,user,clientname,xx,nnn,group#,page#,lock<br>type,<br>SQL STATEMENT...<br><br>The <b>xx</b> field is one of the following isolation levels: <ul style="list-style-type: none"><li>• <i>RR</i> (Read Repeatability)</li><li>• <i>CS</i> (Cursor Stability)</li><li>• <i>RL</i> (Release Locks)</li><li>• <i>RO</i> (Read-only)</li></ul><br>The <b>nnn</b> field is the Lock Timeout Value expressed in number of seconds.<br>The following list shows the lock types: <ul style="list-style-type: none"><li>• <b>Temp S-Lock</b> (temporary shared lock)</li><li>• <b>S-Lock</b> (shared lock)</li><li>• <b>Temp U-Lock</b> (temporary update lock)</li><li>• <b>U-Lock</b> (update lock)</li><li>• <b>Temp X-Lock</b> (temporary exclusive lock)</li><li>• <b>X-Lock</b> (exclusive lock)</li><li>• <b>Increment</b></li></ul><br>Whenever a TIMEOUT or DEADLOCK occurs, the audit displays the SQL statements for each lock holder involved. The user name resulting in a timeout or deadlock is printed last. |

| <b>Record</b>                                                          | <b>Format</b>                                                                                                                                                                                    |
|------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Table access information (queries only) (category 9)                   | 009,datetime,database,username,clientname,tablename                                                                                                                                              |
| Table update information (INSERTS, UPDATES, AND DELETES) (category 10) | 010,datetime,database,username,clientname,INSERT INTO tablename<br>010,datetime,database,username,clientname,UPDATE tablename<br>010,datetime,database,username,clientname,DELETE FROM tablename |

## Performance audit category examples

This section shows examples of all the performance audit categories.

| <b><i>Record</i></b>                                                    | <b><i>Format</i></b>                                                                                                                                                                                                                                                                                                                            |
|-------------------------------------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Connects and disconnects (category 1)                                   | 001,datetime,CONNECT,database,username,clientname,elapsedtime<br>001,datetime,DISCONNECT,database,username,clientname,elapsedtime                                                                                                                                                                                                               |
| SQL Command compilation, execution, retrieval, and storage (category 2) | 002,datetime,COMPILE,database,username,clientname,elapsedtime<br>SQL STATEMENT...<br>002,datetime,EXECUTE,database,username,clientname,elapsedtime<br>SQL STATEMENT...<br>002,datetime,RETRIEVE,database,username,clientname,elapsedtime<br>SQL STATEMENT...<br>002,datetime,STORE,database,username,clientname,elapsedtime<br>SQL STATEMENT... |
| End of transaction (category 3)                                         | 003,datetime,COMMIT,database,username,clientname,elapsedtime<br>003,datetime,ROLLBACK,database,username,clientname,elapsedtime                                                                                                                                                                                                                  |

## Sending a message to an audit file

You can include a message string in an audit file. For example, you may wish to document when you issue a COMMIT in an application, or when a program is waiting for a user response. The message can contain up to 254 characters.

To include a message in an audit, use the AUDIT MESSAGE command. Specify the message and an existing audit operation name. SQLBase writes the message in the audit file(s) associated with this audit name. You can send a message to the audit files for one or all audit operations.

An audit message is indicated by a 998 category number in the audit file, and uses the following record layout:

998,datetime,database,username,clientname,audit message

The following example shows a sample audit message record in the audit file:

998,23-SEP-95 17:00:23.17,ISLAND,SYSADM,WinUser,Example AUDIT message

## Audit file examples

The following examples show both global and performance audit files.

### Global audit

This example shows a sample global audit file.

|                    |   |                                                               |
|--------------------|---|---------------------------------------------------------------|
| Start global audit | ➡ | 000,09-DEC-95 10:47:05.75,GLOBAL,TEST,1,2,3,4,5,6,7,8,9,10    |
|                    |   | 004,09-DEC-95 10:47:05.86,CONNECT,ISLAND,SYSADM,              |
|                    |   | 004,09-DEC-95 10:47:05.86,DISCONNECT,ISLAND,SYSADM,           |
|                    |   | 001,09-DEC-95 10:47:05.92,404,ISLAND,DAVID,                   |
|                    |   | 001,09-DEC-95 10:47:06.03,404,ISLAND,SYSADM,                  |
|                    |   | 010,09-DEC-95 10:47:06.47,ISLAND,SYSADM,,INSERT INTO SYSADM.X |
|                    |   | 004,09-DEC-95 10:47:06.52,CONNECT,ISLAND,DAVID,               |
|                    |   | 002,09-DEC-95 10:47:06.58,1102,ISLAND,DAVID,,X                |
|                    |   | 005,09-DEC-95 10:47:12.34,CREATE TEST                         |
|                    |   | 005,09-DEC-95 10:47:12.40,DEINSTALL TEST                      |
|                    |   | 005,09-DEC-95 10:47:12.40,INSTALL TEST                        |
|                    |   | 005,09-DEC-95 10:47:12.40,DROP TEST                           |
|                    |   | 006,09-DEC-95 10:47:12.89,ISLAND,SYSADM,                      |
|                    |   | 010,09-DEC-95 10:47:12.95,ISLAND,SYSADM,,UPDATE SYSADM.X      |
|                    |   | 009,09-DEC-95 10:47:13.00,ISLAND,SYSADM,,SYSADM.X             |
|                    |   | 010,09-DEC-95 10:47:13.06,ISLAND,SYSADM,,DELETE FROM SYSADM.X |
| Trace message      | ➡ | 998,09-DEC-95 10:47:13.06,ISLAND,SYSADM,,This is The End      |
| Stop this audit    | ➡ | 999,09-DEC-95 10:47:13.11,TEST                                |

### Performance audit

This example shows a sample performance audit file.

The End of Transaction (category 103) marks a COMMIT or ROLLBACK statement. If there is only one user on the system, such as in the following example, all of the category 1 and 2 records contained between the category 103 records are part of the same transaction. However, if there are multiple users on the system, these records are not necessarily part of the same transaction since they can be generated by different users.

Note that the time elapsed field for the category 103 record records the elapsed time since the last category 103 record, not the sum of the elapsed times of the preceding category 101 or 102 records.

|                                                                                                                       |   |                                                                                                                                                                                                                                                                  |
|-----------------------------------------------------------------------------------------------------------------------|---|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Start performance audit                                                                                               | ➔ | 000,30-NOV-95 16:57:59.20,PERFM,perf,1,2,3<br>101,30-NOV-95 16:58:10.02,CONNECT,ISLAND,SYSADM,<br>internal operation,0.20<br>101,30-NOV-95<br>16:58:10.08,DISCONNECT,ISLAND,SYSADM,internal<br>operation,0.00                                                    |
| End of transaction                                                                                                    | ➔ | 103,30-NOV-95 16:58:21.61,ISLAND,SYSADM,,10.75<br>101,30-NOV-95 16:58:21.67,CONNECT,ISLAND,SYSADM,,1.00                                                                                                                                                          |
| End of transaction 2. Note<br>the elapsed time of 2.10 is<br>the amount of time elapsed<br>since transaction 1 ended. | ➔ | 103,30-NOV-95 16:58:23.97,ISLAND,SYSADM,,2.10<br>102,30-NOV-95 16:58:24.14,COMPILE,ISLAND,SYSADM,,0.10,<br>UPDATE X SET A = 1<br>102,30-NOV-95 16:58:48.36,EXECUTE,ISLAND,SYSADM,,22.05,<br>UPDATE X SET A = 1<br>103,30-NOV-95 16:58:48.64,ISLAND,SYSADM,,22.40 |
| Trace message                                                                                                         | ➔ | 998,30-NOV-95 17:15:17.86,SYSADM ANOTHER COMMIT<br>102,30-NOV-95 17:15:17.86,EXECUTE,ISLAND,SYSADM,,0.00,<br>103,30-NOV-95 17:15:33.35,COMMIT,ISLAND,SYSADM,,29.90                                                                                               |
| Stop this audit                                                                                                       | ➔ | 999 23-SEP-95 17:03:33.99,perf                                                                                                                                                                                                                                   |
| Start new audit                                                                                                       | ➔ | 100,23-SEP-95 17:00:13.84,PERFM,perf2,1,2,3                                                                                                                                                                                                                      |



## Chapter 5

# DBA Interfaces

---

This chapter introduces the client programs that you as a DBA can use to administer databases:

- SQLTalk
- Application Programming Interface (API)
- SQLConsole
- SQLBase Server Console
- SQLBase Management Console (SMC)

## Client programs

This section describes Gupta's client programs which enable you to perform DBA tasks and maintain SQLBase databases.

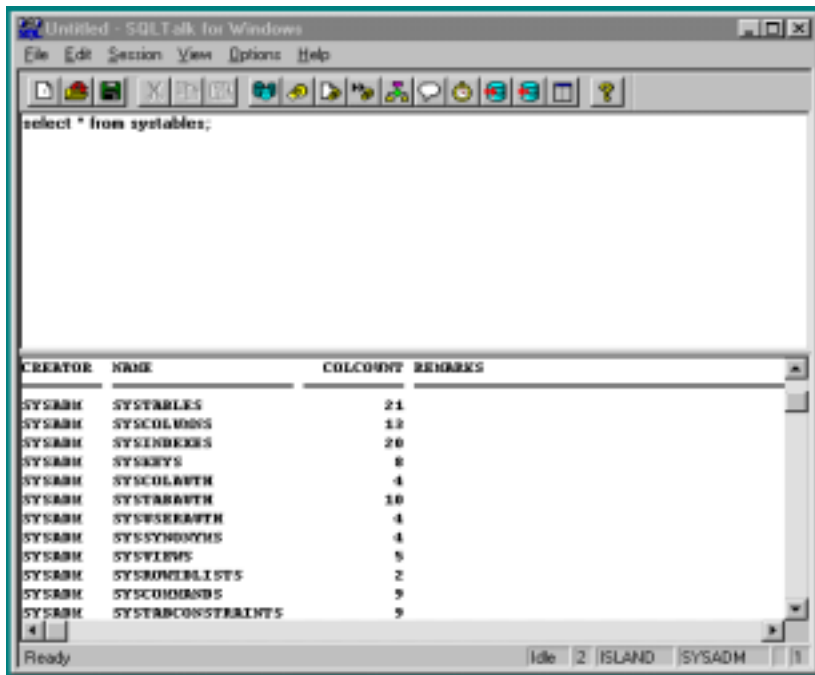
### SQLTalk

SQLTalk is an interactive user interface for SQLBase, offering not only a complete implementation of SQL, but many extensions as well.

SQLTalk runs on the following client platforms:

- Windows 98, ME, NT, 2000 , XP, or Server 2003 using the executable *sqtalk.exe*.

The following example is an excerpt from a SQLTalk session on the Windows 2000 platform.



Note that entering **CONNECT 1** causes SQLTalk to accept the default value of DEMO for the database, SYSADM for the username, and SYSADM for the password.

For more information about SQLTalk, refer to the *SQLTalk Command Reference* or the SQLTalk for Windows Online Help System.

## Application Programming Interface (API)

Gupta's API has a set of functions that you can call to access a database using Structured Query Language (SQL).

SQL (Structured Query Language) can define, manipulate, control, and query data in a relational database. However, SQL is not a programming language and does not provide:

- Procedural logic
- Extensive data typing
- Variables

The API is a language interface that lets you develop a client application that uses SQL. You embed SQL/API functions within your C program, and this enables you to use SQL without giving up the power and flexibility of the programming language.

You can use the SQL/API to write programs that perform DBA operations. This lets you create custom DBA client applications tailored to your specific needs.

The SQL/API is available for all client platforms. Read the *SQL Application Programming Interface Reference* for more information about the SQL/API.

## SQLConsole

SQLConsole for SQLBase is a remote database server tool to perform monitoring and administration tasks. If you are a Windows user, you can use SQLConsole to perform most DBA operations described in this manual for a local SQLBase server or any SQLBase server on your network.

From a single Windows desktop, SQLConsole provides an easy-to-use graphical interface that lets you perform DBA operations without the need for SQL commands. SQLConsole simplifies basic and routine administration tasks, such as creating, installing, deleting, or backing up databases and log files.

SQLConsole offers these capabilities for monitoring and controlling databases and servers:

- Presents important lock, database, process, cursor and activity statistics to the database administrator or SQLBase developer.
- Provides access to a set of very powerful SQLBase functions previously available only to the advanced C programmer.

For example, you may disconnect active sessions, shutdown a database gracefully or terminate a server.

- Provides access to SQLTalk for Windows custom control, an interactive interface for entering SQL and SQLTalk commands, as well as SQLTalk scripts.

SQLConsole also provides these managers:

- **Scheduling Manager:** automates your mission-critical backup cycle.
- **Database Object Manager:** accesses all facets of a server through its graphical user interface.
- **Alarm Manager:** alerts you on state of your server.

Read the *SQLConsole Guide* for details on using SQLConsole for daily database administration and development activities.

## Character-based server interface

The SQLBase server for NetWare has three character-based displays that let you monitor server activity. You can switch among the displays using the **F1**, **F2**, and **F3** function keys.

### Server Status display screen (F1)

SQLBase Server NLM for NetWare Systems 8.0.010:23 \*  
Copyright(C) Gupta Technologies LLC 1985-2002  
All Rights Reserved

FUNCTION KEYS

F1-Server StatusF2-Process Activity  
F3-System ActivityF4-Security PasswordEsc-Exit Server

SERVER STATUS: S207

| CLIENT NODE | USER NAME | DATABASE | STATUS |
|-------------|-----------|----------|--------|
| D65D0E8C600 | JIMP      | ACCTPAY  | IDLE   |
| 5765768C600 | KARENK    | DEMO     | IDLE   |
| E3CB2A8C600 | KAMALI    | INVEN    | IDLE   |

DATABASES  
DEMO  
ACCTPAY  
INVEN  
SALES

*Server Status display of SQLBase Server for NetWare*

**Banner.** The banner tells you the name and version of the product currently running — in this case, SQLBase Server 8.0.0 for NetWare.

**Function Keys.** The function keys section serves as a guide for getting to the rest of the displays and bringing down the server.

**Server Status.** By default, SQLBase randomly generates a name for the server (in this case, S207) if you do not configure a *servername* keyword in the server's configuration file (SQL.INI).

The Server Status section lists both the databases on the server and the clients connected to those databases:

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Client Node | <p>Identifies the client node associated with this process. This first appears as a 0 (zero) and then changes to an identifying number once the connection is successful.</p> <p>Based on the platform, SQLBase has various ways of generating this value:</p> <p>For a Windows client, SQLBase displays a randomly-generated number.</p> <p>For a client whose configuration file (SQL.INI) includes a <i>timebased</i> keyword, SQLBase displays a timestamp.</p> <p>If a client connecting to the server has a <i>clientname</i> keyword configured in its SQL.INI file, that value overrides any of the above defaults.</p> <p>Read <i>Chapter 3, Configuration: SQL.INI and the Registry</i> for more information about the <i>clientname</i> keyword.</p> |
| User Name   | The name of the user associated with this process.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |
| Database    | The database being accessed.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
| Status      | Whether the process is active or idle. Active means that the process is in the server's run queue.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |

By default, the server displays only the Client Node, User Name, and Database information. You can press the plus (+) key on the keypad to display the Status column, although this may affect system performance. Press the minus (-) key to remove the Status column.

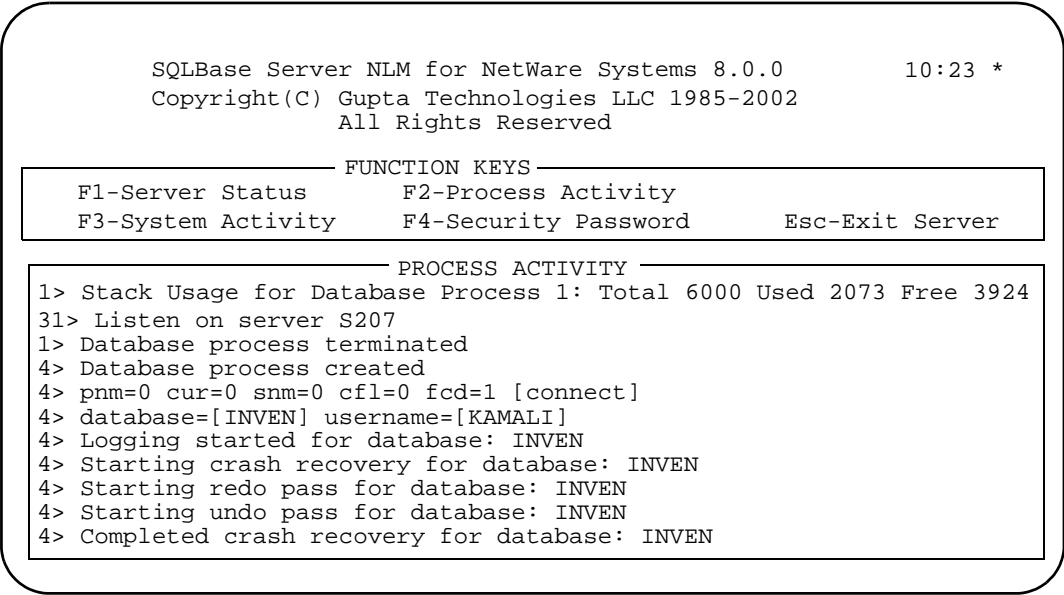
**Databases.** This window lists the databases on the server. SQLBase gets this information from the *dbname* keywords in the configuration file (SQL.INI).

**Busy Indicator.** The server displays a busy indicator (flashing asterisk) in the top right corner of the screen when the server is active. When the busy indicator is *not* displayed, the server is idle. The asterisk is not displayed at print level 0 (see next section).

SQLBase updates the Server Status display every time a client connects to, or disconnects from, a database. When clients are not frequently connecting and

disconnecting, SQLBase refreshes the display periodically. The refresh frequency decreases over time if the server is idle. Because of this, the displayed time may not always be accurate.

## Process Activity display screen (F2)



### *Process Activity display of a SQLBase Server for NetWare*

**Banner.** The banner tells you the name and version of the product currently running — in this case, SQLBase Server 8.0.0 for NetWare.

**Function keys.** The function keys section serves as a guide for getting to the rest of the displays and bringing down the server.

**Process Activity.** The Process Activity display shows information about the tasks the server is performing.

If you have set your message display at level 2 or higher, the database processes information displays the following parameter values:

| Field | Explanation                              |
|-------|------------------------------------------|
| PNM   | The client process number.               |
| SNM   | The client communication session number. |
| CFL   | A flag byte value                        |
| FCD   | Function code                            |

**Message levels.** Press the plus (+) key on the keypad to display additional levels of messages. The minus (-) key returns you to the previous level. The five message levels are:.

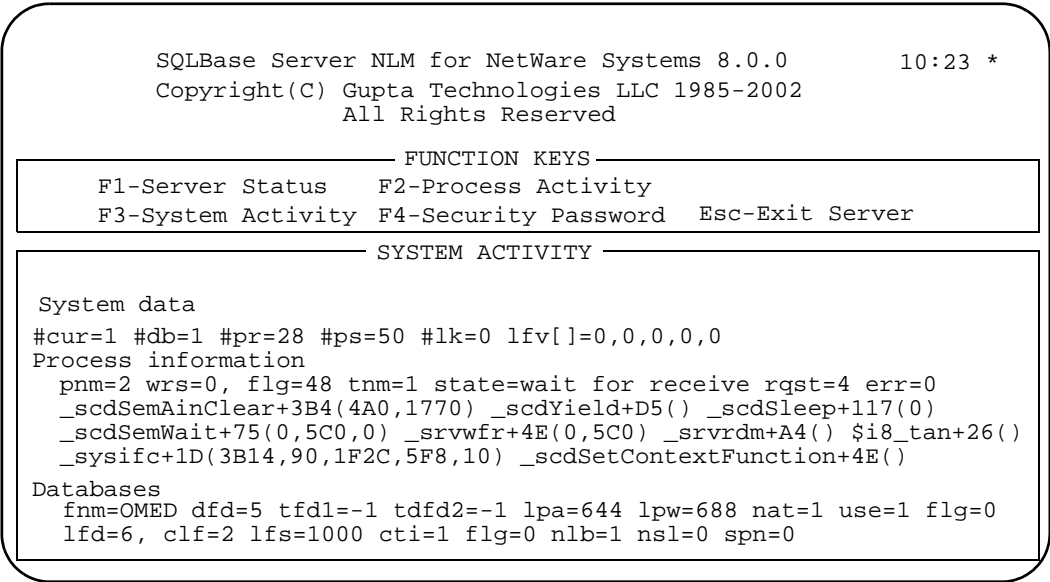
| Message Level | Description                                                                                                                                                                                                     |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 0             | The installation of the server and database names on the network, as well as each database process's creation and termination.                                                                                  |
| 1             | All the information available at level 0 plus SQL statements and error numbers. (Note this includes viewing messages for the LOAD and UNLOAD commands.)                                                         |
| 2             | All the information available at level 1 plus the compile and execute processing steps, the ability to see the processing of individual cursors within a database process, and internal processing information. |
| 3 and 4       | All the information available at level 2 plus the fetch processing step and bind variable data.                                                                                                                 |

**Note:** If a database is encrypted, SQL statements will not be displayed, even though the message level may be 1 or higher.

**Logging.** Configure the *log* keyword in the server's configuration file (SQL.INI) to write the Process Activity messages to a file. These messages are helpful during system development, but they are not meant to be logged in a production system. If you are logging database messages, the log file can get quite large. Also, the server has to do additional work to write to the log file which affects system performance.

If you press **Alt-T**, SQLBase timestamps each message. Press **Alt-T** again to toggle off the timestamping feature.

System Activity display screen (F3)



System Activity display of a SQLBase Server for NetWare

**Banner.** The banner tells you the name and version of the product currently running — in this case, SQLBase Server 8.0.0 for NetWare.

**Function keys.** The function keys section serves as a guide for getting to the rest of the displays and bringing down the server.

**System Activity.** The System Activity display shows system information about internal server activity.

The System Activity display has three sections:

- System data
- Process information
- Databases

Initially, no system activity is shown. Pressing the plus key (+) once shows the System data section followed by the Databases section. Pressing the plus key again shows System data, then Process information, then Databases. Pressing the minus key (-) at this point reverses the process - the screen will show System data followed by Databases. Pressing the minus key again will hide all system activity.

## System data

This section contains information about the server.

| Field | Explanation                                                                                                                                                                                                                    |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cur   | The number of cursor slots allocated since the server started.                                                                                                                                                                 |
| db    | The number of activated databases since the server started.                                                                                                                                                                    |
| pr    | The maximum number of users allowed on the server. You can set this limit in the server's <code>SQL.INI</code> file with the <code>users</code> keyword. It can be higher than the value specified because of system overhead. |
| ps    | The number of cache pages on the server. You can set this value in the server's <code>SQL.INI</code> file with the <code>cache</code> keyword.                                                                                 |
| k     | The number of lock entries on the server. You can set this value in the server's <code>SQL.INI</code> file with the <code>locks</code> keyword. The default value of 0 means there is no limit on the number of locks.         |
| fv    | The system resources held. A non-zero value means that the resource is being held and represents the usage; a value of zero means that the resource is not being held.                                                         |

## Process information

This section contains information about each process:

| Field | Explanation                                                                                                                                                                                                                                                                          |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| nm    | The process number.                                                                                                                                                                                                                                                                  |
| wrs   | The resource for which the process is waiting.                                                                                                                                                                                                                                       |
| lg    | Flags.                                                                                                                                                                                                                                                                               |
| nm    | The transaction number.                                                                                                                                                                                                                                                              |
| state | A message indicating the process's activity, such as:<br>"waiting for request"<br>"performing request"<br>"sending response"<br>"handling error"<br>"wait for IO lock"<br>"wait for page"<br>"request done"<br>"waiting for received" (waiting to receive something from the client) |
| qst   | The last request received.                                                                                                                                                                                                                                                           |

| Field | Explanation                          |
|-------|--------------------------------------|
| err   | The return code of the last request. |

It then lists the system locks held by the current process:

| Code | Explanation                       |
|------|-----------------------------------|
| ST   | A single thread system lock.      |
| KI   | A keyboard input system lock.     |
| CNC  | A connect/disconnect system lock. |

Is also shows the system locks that the current process is for. This part is absent if the process is not waiting for a system lock.

Finally, it shows the resource (page) the current process is waiting for. This part is absent if the process is not waiting for a resource:

| Field | Explanation                                                                                                                                                                              |
|-------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wlm   | The lock mode the current process is requesting:<br>S = Shared<br>U = Update<br>X = Exclusive<br>I = Increment<br><br>When a T is prefixed to S, U, or X, it signifies a temporary lock. |
| wdb   | The database number.                                                                                                                                                                     |
| wgp   | The group number of the page.                                                                                                                                                            |
| wpg   | The page number.                                                                                                                                                                         |

Databases

This section contains information about the databases that the server listens on:

| Field          | Explanation              |
|----------------|--------------------------|
| nm             | The database name.       |
| dfd, fd1, dfd2 | The file descriptor.     |
| pa             | The last page allocated. |

| Field                        | Explanation                                              |
|------------------------------|----------------------------------------------------------|
| lpw                          | The last page written (the database file size in pages). |
| nat                          | The number of active transactions.                       |
| use                          | The number of connected cursors.                         |
| lg                           | Flags.                                                   |
| fd                           | The log file descriptor.                                 |
| clf                          | The current log file number.                             |
| lfs                          | The log file size in kilobytes.                          |
| cti                          | The checkpoint time interval in minutes.                 |
| flg                          | Flags.                                                   |
| nlb                          | The next log to backup.                                  |
| nsl                          | The next log to backup (after a BACKUP SNAPSHOT).        |
| spn                          | The transaction span limit (0 means no limit).           |
| cpp, cpl, ftp, fap, fbl, lbl | Log pointers.                                            |

The last part of the Databases section is a dump of the lock table. For example:

```
1 S004:006:006
```

The '1' indicates the transaction number that owns the lock entry. In some cases, this is the process number for special locks.

The 'S' means that it is a shared lock. This can also be:

X = Exclusive

U = Update

I = Increment

When T is prefixed to S, U, or X, it signifies a temporary lock.

The '004' is the group to which the lock belongs. SQLBase divides pages in a database into groups.

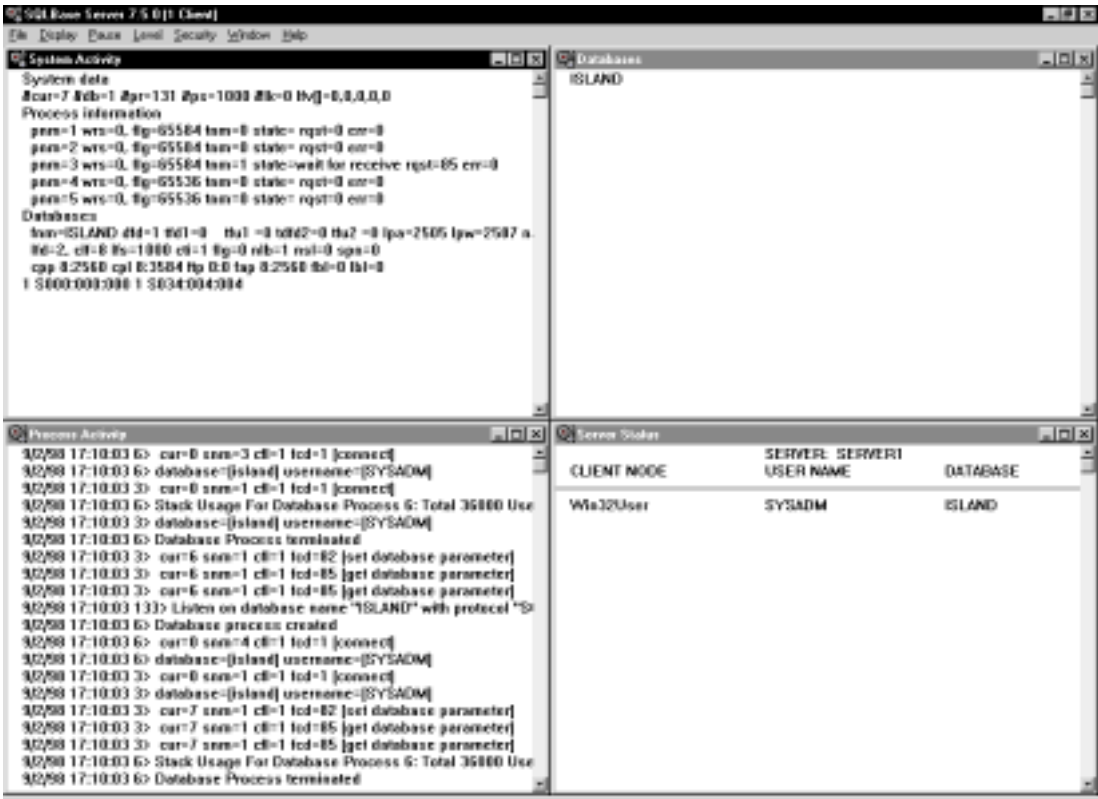
The first '006' is the starting page number and the second '006' is the ending page number.

NetWare specifics

If the file server is not currently displaying one of the SQLBase server display screens, press **Ctrl** and **Esc** and select SQLBase from the task list.

SQLBase Server Console

This section describes the Graphical User Interface (GUI) to SQLBase servers for Windows NT (4.0 and later), 2000, XP, and Server 2003.



The MDI parent window appears when you start the database server. The first time you start SQLBase, only the MDI parent window is visible. To see the MDI children, choose **Display, All**.

The MDI parent window has four MDI children:

- Server Status
- Databases
- Process Activity
- System Activity

These windows are described in the sections after the next subsections about the SQLBase Server Console's menus.

## Saving configuration information

If you manipulate any of the server windows to a customized setting, you can save this configuration. You can also save specific display levels and timestamp settings. To save these configurations, use the *Save Settings* option of the server's **File** menu.

## Scrolling and pausing

You can scroll or pause the information displayed on any of the sub-windows. Be aware that scrolling does not prevent SQLBase from writing new output to a sub-window. For example, if you are scrolling backwards and SQLBase has new output to write to the display, it will scroll forward to the end of the output and append the new information to the display.

To prevent SQLBase from writing new output to a window and interrupting you while scrolling, select the **Pause** menu command, and then select the name of the window to pause. A check mark indicates that the window is paused.

When you are finished scrolling, select the name of the window to un-pause from the **Pause** menu command again. The check mark disappears, and output resumes.

Because all of the sub-windows (except the Process Activity window) display snapshots of the server's status at any particular time, when you discontinue **Pause**, SQLBase updates the display with the latest information.

However, the Process Activity window displays a log, not a snapshot. It contains a record of the server's activity over time. When you pause the display of this window, SQLBase buffers the output until you click **Pause** again; this un-pauses the displays. The limit to the amount of information that SQLBase can buffer is 500 lines. When the buffer becomes full, SQLBase "wraps" back to the beginning of the buffer and starts overwriting data. This means that the data being overwritten is lost. If you need to track large amounts of Process Activity data, you should log the data to a file using the **File** menus' **Open Log** menu command.

There are seven menus in SQLBase Server Console:

- File
- Display

- Pause
- Level
- Security
- Window
- Help

The following paragraphs describe these menus.

File

The **File** menu contains commands that control logging, GUI configuration, and exiting the server.

|                |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
|----------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Open Log       | <p>Opens a file and begins logging the Process Activity display to this file.</p> <p>This command presents you with the <i>File Open</i> dialog box that lets you select an existing file, create a new file, or accept the default of <i>sqllog.txt</i> in the current directory. If the file you specify already exists, SQLBase asks you to confirm that you want to overwrite the file.</p> <p>This command is equivalent to configuring the <i>log</i> keyword in the server’s SQL.INI file, or using SQLTalk’s SET ACTIVITYLOG command.</p> |
| Close Log      | <p>Closes the process activity log file.</p>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      |
| Save Settings  | <p>Saves the current SQLBase Server Console GUI-related settings to the server’s SQL.INI file. These settings include the:</p> <ul style="list-style-type: none"><li>• Position and size of the main window and all sub-windows</li><li>• Display level</li><li>• Timestamp option setting</li></ul> <p>You can also use this option to save the server screen appearance between server launches.</p> <p>For more about the specific keywords used in SQL.INI, read <i>Configuration</i> on page 5-21.</p>                                       |
| Clear Settings | <p>Removes any SQLBase Server Console GUI-related settings from the server’s SQL.INI file.</p> <p>The next time you bring up the server, it uses the system defaults.</p>                                                                                                                                                                                                                                                                                                                                                                         |

|      |                                                                                                                                                                                                                                                                                                                                             |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Exit | <p>Exits SQLBase Server.</p> <p>If there are users connected to databases on the server, you are asked to confirm that you want to shut down the server. If you shut down the server while users are connected, all transactions in progress are rolled back.</p> <p>This menu item is disabled if SQLBase is running as an NT service.</p> |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

## Display

The **Display** menu contains the commands that control the display of the sub-windows. If a sub-window is displayed, a check mark appears next to it on the menu.

|                  |                                                                |
|------------------|----------------------------------------------------------------|
| Server Status    | Toggles the display of the Server Status window on and off.    |
| Databases        | Toggles the display of the Databases window on and off.        |
| Process Activity | Toggles the display of the Process Activity window on and off. |
| System Activity  | Toggles the display of the System Activity window on and off.  |
| All              | Displays all of the currently-closed windows.                  |
| None             | Closes all of the currently-displayed windows.                 |

## Pause

The Pause menu controls whether each subwindow is paused. If a subwindow is paused, a check mark appears next to it on the menu.

## Level

The **Level** menu contains commands that control the level of detail displayed on the Process Activity window. Only one of the five levels (0-4) can be active at a time. The currently-active level is indicated by a check mark next to it on the menu. These menu selections are equivalent to using SQLTalk's SET PRINTLEVEL command.

If timestamps are enabled, a check mark appears next to the corresponding menu selection. This menu selection is equivalent to using SQLTalk's SET TIMESTAMP ON/OFF command.

|         |                                                                                                                                         |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------|
| Level 0 | Displays the installation of the server and database names on the network, as well as each database process's creation and termination. |
|---------|-----------------------------------------------------------------------------------------------------------------------------------------|

|                 |                                                                                                                                                                                                                          |
|-----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Level 1         | Displays all the information available at level 0 plus SQL statements and error numbers.                                                                                                                                 |
| Level 2         | Displays all the information available at level 1 plus the compile and execute processing steps, the ability to see the processing of individual cursors within a database process, and internal processing information. |
| Level 3         | Displays all the information available at level 2 plus the fetch processing step and bind variable data.                                                                                                                 |
| Level 4         | Displays all the information at level 3 plus database page locks.                                                                                                                                                        |
| Increase Level  | Increases the display level.                                                                                                                                                                                             |
| Decrease Level  | Decreases the display level.                                                                                                                                                                                             |
| Show Timestamps | Toggles the display of timestamps on the Process Activity window on and off.                                                                                                                                             |

---

**Note:** If a database is encrypted, SQL statements will not be displayed, even though the message level may be 1 or higher.

---

### Security

The **Security** menu provides options for specifying and setting the server security password. For more, read *Server security password* on page 7-17.

|                          |                                                           |
|--------------------------|-----------------------------------------------------------|
| Set Security Password    | Initially sets or specifies the server security password. |
| Change Security Password | Changes the server security password.                     |

### Window

The **Window** menu provides options for displaying open sub-windows:

|         |                                                                                                                             |
|---------|-----------------------------------------------------------------------------------------------------------------------------|
| Tile    | Displays the sub-windows side-by-side so that the maximum amount of information is displayed.                               |
| Cascade | Displays the sub-windows layered on top of one another so that only their title bars are visible (except the top-most one). |

### Help

The **Help** menu provides information about the server program: its name, version number, and copyright.

## Server Status

This sub-window displays the server name on the top line, and contains the following information for each connection:

|             |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Client Node | Identifies the client node. This first appears as a 0 and then changes to an identifying number when the connection is successful: <ul style="list-style-type: none"> <li>• For a Windows client application, this is a randomly-generated number.</li> <li>• If a client communicating via NetBIOS has a <i>timebased</i> keyword configured, this is a timestamp.</li> <li>• If a client has a <i>clientname</i> keyword configured, that value displays instead of any of the above.</li> </ul> |
| User Name   | Identifies the name of the user.                                                                                                                                                                                                                                                                                                                                                                                                                                                                   |
| Database    | Identifies the database to which each client is connected.                                                                                                                                                                                                                                                                                                                                                                                                                                         |
| Status      | Optional. You can press <b>Ctrl+</b> to display this field which shows you if the client is active or idle. Displaying the Status field can affect system performance, so press <b>Ctrl-</b> to remove this field from view when you do not need to see it.<br><br>A status of <b>ACTIVE</b> means that the client process is in the server's run queue. Otherwise, the status displays as <b>IDLE</b> .                                                                                           |

SQLBase updates the display every time a client node connects or disconnects.

## Databases

This sub-window lists the databases on which the server listens. The database server gets this information from the *dbname* keywords configured in its SQL.INI file.

## Process Activity

This sub-window displays information about tasks that the server is performing.

Press **Ctrl+** to display additional levels of messages. Press **Ctrl-** to reduce the level of messages displayed.

See the **Level** menu explanation for detailed information about each level of display.

Configuring the *log* keyword in the server's SQL.INI file, or selecting the **File** menu's **Open Log** command, writes the information displayed in this window to a specified file. This information is helpful during system development, but it is not meant to be

logged in a production environment. Logging incurs overhead, and the log file can become large very quickly if your site is an active one.

## System Activity

This sub-window displays system information about internal activity. The display has three sections:

- System data
- Process information
- Databases

### System Data

This section contains data about the server:

| Field | Description                                                                                                                                                                                                              |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| cur   | The number of cursor slots allocated since the server started.                                                                                                                                                           |
| db    | The number of activated databases since the server started.                                                                                                                                                              |
| pr    | The maximum number of users allowed on the server. You can set this limit in the server's <code>SQL.INI</code> file with the <i>users</i> keyword. It can be higher than the value specified because of system overhead. |
| ps    | The number of cache pages on the server. You can set this value in the server's <code>SQL.INI</code> file with the <i>cache</i> keyword.                                                                                 |
| lk    | The number of lock entries on the server. You can set this value in the server's <code>SQL.INI</code> file with the <i>locks</i> keyword. The default value of 0 means that there is no limit on the number of locks.    |
| lfv   | The system resources held. A non-zero value means that the resource is being held and represents the usage; a value of zero means that the resource is not being held.                                                   |

### Process Information

This section contains the following information about each process:

| Field | Description                              |
|-------|------------------------------------------|
| pnm   | The process number.                      |
| wrs   | The resource the process is waiting for. |
| flg   | Flags.                                   |

| Field | Description                                                                                                                                                                                                                                                                         |
|-------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| tnm   | The transaction number.                                                                                                                                                                                                                                                             |
| state | A message indicating the process' activity, such as:<br>"waiting for request"<br>"performing request"<br>"sending response"<br>"handling error"<br>"wait for IO lock"<br>"wait for page"<br>"request done"<br>"waiting for received" (waiting to receive something from the client) |
| rqs   | The last request received.                                                                                                                                                                                                                                                          |
| err   | The return code of the last request.                                                                                                                                                                                                                                                |

It then lists the system locks held by the current process:

| Code | Description                   |
|------|-------------------------------|
| ST   | A single thread system lock.  |
| KI   | A keyboard input system lock. |
| CNC  | A disconnect system lock.     |

It also shows the system locks that the current process is waiting for. This part is absent if the process is not waiting for a system lock.

Finally, it shows the resource (page) the current process is waiting for. This part is absent if the process is not waiting for a resource:

| Field | Description                                                                                                                                                                          |
|-------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| wlm   | The lock mode the current process is requesting:<br>S = Shared<br>U = Update<br>X = Exclusive<br>I = Increment<br>When a T is prefixed to S, U, or X, it signifies a temporary lock. |
| wdb   | The database number.                                                                                                                                                                 |
| wgp   | The group number of the page.                                                                                                                                                        |
| wpgw  | The page number.                                                                                                                                                                     |

## Databases

This section contains information about the databases that the server listens on:

| Field                        | Description                                              |
|------------------------------|----------------------------------------------------------|
| fnm                          | The database name.                                       |
| dfd,tfid1 tfid2              | The file descriptor.                                     |
| lpa                          | The last page allocated.                                 |
| lpw                          | The last page written (the database file size in pages). |
| nat                          | The number of active transactions.                       |
| use                          | The number of connected cursors.                         |
| flg                          | Flags.                                                   |
| lfd                          | The log file descriptor.                                 |
| clf                          | The current log file number.                             |
| lfs                          | The log file size in kilobytes.                          |
| cti                          | The checkpoint time interval in minutes.                 |
| flg                          | Flags.                                                   |
| nlb                          | The next log to backup.                                  |
| nsl                          | The next log to backup (after a BACKUP SNAPSHOT).        |
| spn                          | The transaction span limit (0 means no limit).           |
| cpp, cpl, ftp, fap, fbl, lbl | Log pointers.                                            |

The last part of the Databases section is a dump of the lock table. For example:

```
1 S004:006:006
```

The ‘1’ indicates the transaction number that owns the lock entry. In some cases, this is the process number for special locks.

The ‘S’ means that it is a shared lock. This can also be:

- X = Exclusive
- U = Update
- I = Increment

When a T is prefixed to S, U, or X, it signifies a temporary lock.

The '004' is the group to which the lock belongs. SQLBase divides pages in a database into groups.

The first '006' is the starting page number and the second '006' is the ending page number.

## Configuration

The dbntrsv.gui section of SQL.INI contains settings for the GUI of SQLBase Server Console.

In this section, you can configure these keywords:

|                  |                |
|------------------|----------------|
| <i>dbwin</i>     | <i>statwin</i> |
| <i>displevel</i> | <i>syswin</i>  |
| <i>mainwin</i>   | <i>procwin</i> |

Use the Save Settings and Clear Settings options of the SQLBase Server Console's **File** menu to configure these keywords. Read *Chapter 3, Configuration: SQL.INI and the Registry* for detailed information about the keywords.

## SQLBase Management Console (SMC)

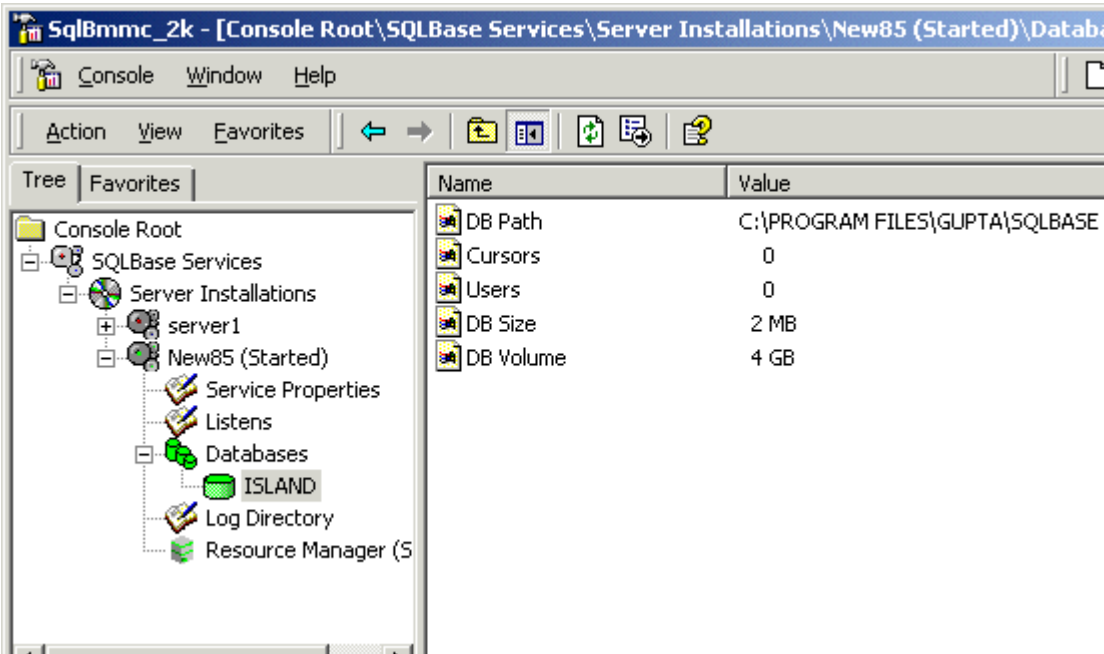
---

**Note:** SMC allows monitoring of SQLBase to any user who has local access to the system. For this reason, you may need to take precautions against unauthorized use of the SQLBase Server.

---

SQLBase Management Console (SMC) allows you to monitor SQLBase Server, to start and stop the server, and to change some server configuration values. SMC is started from the Gupta program group.

It is implemented as a “snap-in” to the Microsoft Management Console tool that is included with Windows. SMC's GUI interface provides live server monitoring and displays information in an easy-to-view tree format.



You can expand the nodes in the SMC left pane to see more detail. When you left-click on a node, you get status information for that node in the right pane, as shown for database ISLAND in the example above.

SMC information displays

Here is a summary of the information presented by each type of node in SMC:

Service Properties Node

Platform and version information: SMC displays both the SQLBase version and Operating System version in the right pane when the Service Properties node is active in the left pane.

Listens node

The Listens Node displays the network protocols that SQLBase Server is using.

Databases node

SQLBase Management Console is set to automatically monitor SQLBase databases:

Database Name

**Database Path:** Operating System path to the database.

**Database Size:** allocated space for the database.

## Database node

**Cursors:** Number of cursors connected to the database.

**Database Size:** Actual database file size

**Database Volume Size:** Free space on the database volume.

**Users:** Number of users connected to the database and user names.

## Resource Manager node

This node displays statistics for SQLBase Resource Manager, which controls COM+ transactions that involve the SQLBase server.

Connects

Disconnects

Enlists

Prepares

Committed

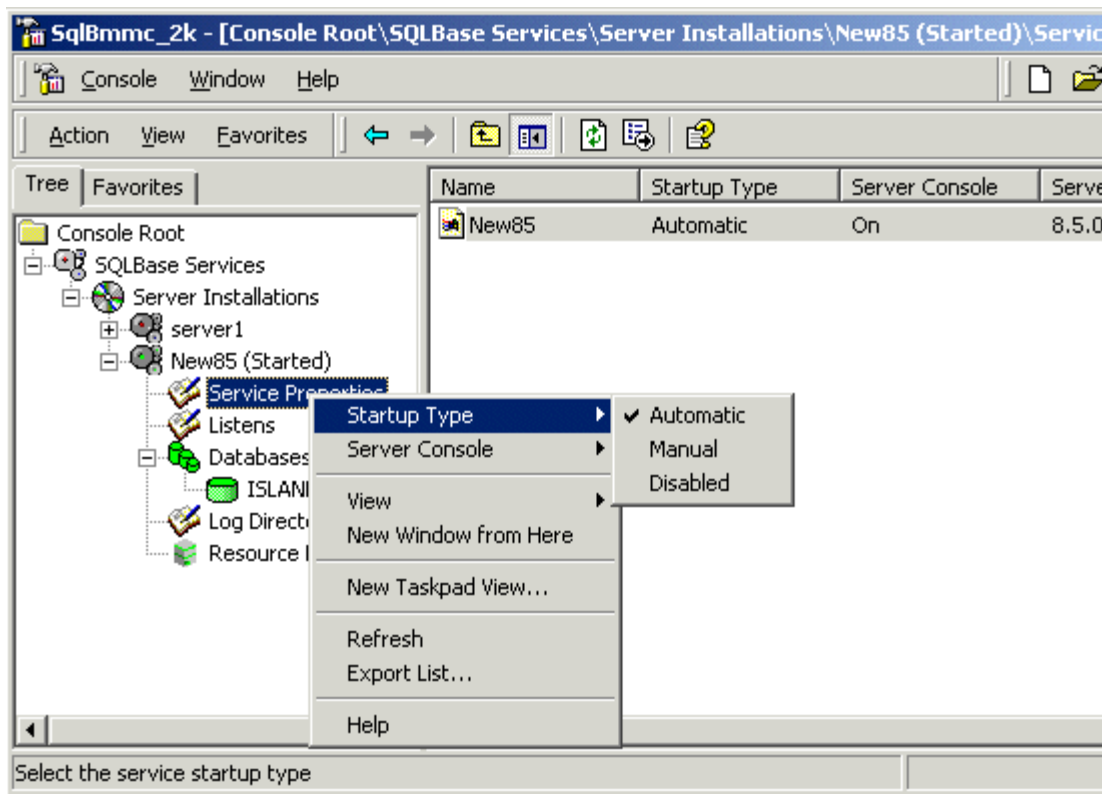
Aborted

Recovery enlists

## SMC actions are performed by context menus

Clicking on the nodes in the SMC left pane provides detailed informational displays. But if you wish to perform configuration or operational actions, you will need to

right-click on a node and choose from context menus. Each type of node presents different context menus.



In the example above, the context menu for Service Properties is shown.

There are some general context menu items found on most nodes, and others that are specific to certain types of nodes. The general context menu items are:

**New Window from Here:** reconfigures the SMC display to show the currently selected node, and its descendants, onln.

**Refresh:** updates the information display in the right pane for the currently selected node.

**Help:** displays online help.

Here is a summary of the various types of nodes displayed in SMC, and the specific actions available from their context menus:

Server Installations node:

**Active Installation** allows you to choose which server installation that you want

to make active.

**Note:** The selected installation will become active only after the server is restarted next time. A message box will be displayed stating this information and to restart the server for changes to take effect.

### Server node:

**Stop:** If SQLBase is running as a service or an application program, click this icon to stop SQLBase. A warning message displays if users are currently connected. When you shut down SQLBase as a service using SMC, the shutdown is unconditional. For more information, read Shutting down SQLBase Server for Windows. When you shut down SQLBase as an application program, shut down occurs when no users are left. SQLBase will not accept any more connections during this time.

**Start Service:** (This option is disabled if SQLBase is already running.) SQLBase begins running as a Windows service.

**Start Program:** (This option is disabled if SQLBase is already running.) SQLBase begins running as an application program.

### Service Properties node:

**Startup Mode:** determines how SQLBase starts. Submenu options are:

Manual: Allows you to start SQLBase as a service (after starting Windows) using SMC or the Windows Services Manager.

Automatic: Allows you to start SQLBase as a service automatically when Windows starts up.

Disabled: Allows you to start SQLBase as an application only.

**Server console setting:** determines whether the SQLBase GUI and icon are visible to users. Submenu options are:

Off: The SQLBase GUI and icon are not visible to users.

On: The SQLBase GUI and icon is visible and enabled for user interaction.

If SQLBase is running as a service, the SQLBase Exit menu item is disabled to prohibit users from shutting down SQLBase.

### Database node:

Refresh: updates the statistics shown for a specific database.

### Resource Manager node:

**Stop:** Resource Manager presents a message box asking for confirmation, then stops running.

**Start RM as Service:** Resource Manager presents a message box asking for confirmation, then begins running as a service.

**Start RM as Service w/ Parameters:** Resource Manager presents a message box asking for confirmation, presents a dialog box allowing you to enter parameters (see below), then begins running as a service.

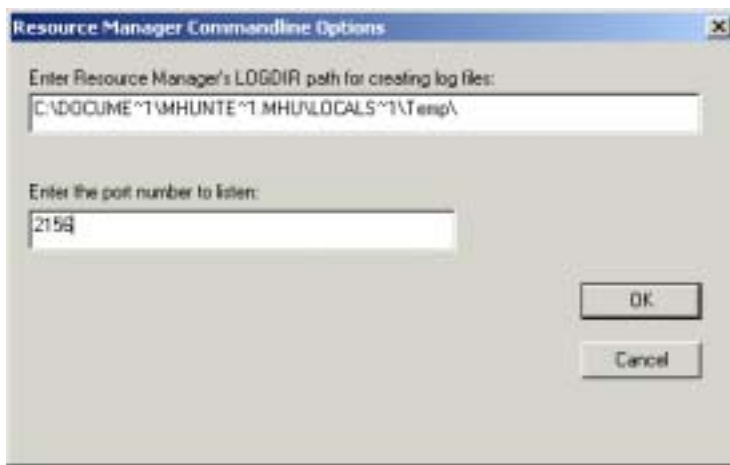
**Start RM as a Program:** Resource Manager presents a message box asking for confirmation, then begins running as an application program.

**Start RM as Service w/ Parameters:** Resource Manager presents a message box asking for confirmation, presents a dialog box allowing you to enter parameters (see below), then begins running as an application program.

---

**Note:** The menu items to start Resource Manager as a program will not be enabled by default. They will be enabled only when Resource Manager starting as a service has been disabled.

---



Parameters are presented in a dialog box that displays the default values for log file location and for the TCP/IP port number to which Resource Manager will listen. You can modify these values if you wish, then click OK to start Resource Manager.

## Chapter 6

# DBA Operations

---

This chapter steps you through some of the tasks you will perform as a DBA. It also indicates the methods available to perform the task. This chapter includes the following tasks:

- Starting and stopping SQLBase
- Creating and deleting a database
- Deinstalling and re-installing a database
- Loading and unloading a database
- Creating a read-only database
- Upgrading your SQLBase software
- Partitioning a database
- Maintaining a partitioned database
- Reorganizing a database
- Checking database integrity

---

**Note:** If you use Windows NT (4.0 and later), 2000, XP, and Server 2003, you can also use Gupta's SQLConsole to perform most DBA operations listed in this chapter. From a single Windows desktop, SQLConsole provides an easy-to-use graphical interface that lets you perform DBA operations, like database load and unload operations without the need for SQL commands. Read the *SQLConsole Guide* for details.

---

## Starting and stopping SQLBase

This section shows you how to start and stop each of the SQLBase single-user database servers and multi-user database servers.

You must start a server before you can access a database from a client application.

Make sure that the files listed below are in a directory where SQLBase can find them. The best way to ensure this is to keep them in the same directory as the rest of the SQLBase software (C:\Program Files\Gupta, by default).

- error.sql
- message.sql
- SQL.INI
- country.sql
- main.ini

## NetWare

This section describes how to start and stop the SQLBase Server for NetWare for the 4.11 and 5.0 versions. You must start the database server before clients can access a database.

### Starting

SQLBase Server for NetWare is a NetWare Loadable Module (NLM) that runs on Novell's NetWare server operating system. An NLM is a program that you can load into or unload from server memory while the server is running. When loaded, an NLM is part of the NetWare operating system. When unloaded, an NLM releases the memory and resources that were allocated for it.

### Loading the SQLBase Server for NetWare

To load SQLBase Server for Netware, use the LOAD command. The following examples illustrate how to load a 4.11 and 5.0 multi-user server. Note that you must substitute your server name. The following examples assume that `nlm:\Gupta` is the NetWare volume you installed into.

To load the 4.11 or 5.0 multi-user server:

```
load nlm:\Gupta\dbnwsrv
```

The default file system driver is dfd. This driver loads automatically, unless you specify otherwise. To use dfs instead of dfd, you would add dfs to the end of the load statement. For example:

```
load nlm:\Gupta\dbnwsrv dfs
```

The database server is removed from memory when the computer is switched off or rebooted. You must start the database server after each startup or reboot.

## Stopping

You should stop the SQLBase server before turning the database server computer off. To stop the SQLBase server:

1. Press the **Esc** key.
2. If there are users connected to the server, a dialog box appears asking you to confirm that you want to stop the server. Press **Y** to bring the server down gracefully.

After stopping SQLBase, the server unloads itself and associated SQLBase support NLMs.

---

**Note:** You cannot unload the SQLBase NLM from the console prompt of the NetWare server; you must always use the ESC key to bring down the server. Once loaded, the SQLBase NLM becomes a NetWare resource and must be closed before it can be unloaded.

---

## Windows

This section describes how to start and stop the SQLBase Server for 32-bit Windows environments. You must start the database server before clients can access a database.

Both the Windows 98/ME and Windows NT/2000/XP/2003 servers have a graphical user interface (GUI) that is described in *SQLBase Server Console* on page 5-12.

## Starting as a program

To start the SQLBase server as a program, do one of the following:

- Configure the server to start automatically upon a client request. (See keyword AUTOSTARTSERVERPATH in chapter 3.)
- If an icon exists for the server program, double click on it.
- Use SQLBase Management Console, locate the server name to be started, right-click, and choose Start as Program.
- Otherwise, select **File, Run** or **Start, Run** and enter **dbnwsrv**. For example, the following command starts the SQLBase Server for Windows NT that was installed in the \Gupta directory:

```
"c:\Gupta\dbntrsrv" "ini=c:\Gupta\myconfig.ini"
```

Click **OK**.. (In this example, the *ini* command line argument forces the server to use the configuration file *myconfig.ini*. This argument is optional, not required.)

You should stop the SQLBase server before turning the database server computer off. To do so, if the server is running as a program, select **Exit** from the **File** menu of the SQLBase menu bar, or use SQLBase Management Console, right-click on the server name, and choose Stop.

## Starting as a service

To start the SQLBase server as a Windows service, do one of the following:

- Use SQLBase Management Console, locate the server to be started, right-click, and choose Start as Service.
- Use the Windows Services display, locate the Gupta SQLBase servername entry, right-click and choose Start.
- On a command line, enter  

```
NET START "Gupta SQLBase servername"
```

The quotes are required. *servername* represents the actual name of your particular SQLBase server. You can also add "*ini=path\file*" to the end of the command to force the use of a particular configuration file.

## Stopping

If SQLBase is running as a Windows service, use SQLBase Management Console or the Windows Services display to stop it. You may also use this command from the command line: 

```
NET STOP "Gupta SQLBase servername"
```

.

You must start the database server after each startup or reboot, unless it has been configured to start automatically as a service when Windows starts. You can configure this option in SQLBase Management Console.

## Setting configuration keywords

You can set the value of some configuration keywords (such as *dbdir*) when you start a server as a program. Specify a keyword and its value on the command line to override a SQL.INI configuration entry.

For example, the following command starts SQLBase Server for NT and specifies the database home directory (*dbdir*) as *d:\test\db*. This overrides a *dbdir* configuration entry in the [dbntrsrv] section of the SQL.INI file.

```
dbntrsrv dbdir=d:\test\db
```

You can specify more than one keyword on the command line, each separated by a space, for example:

```
dbntsrv dbdir=d:\test\db log=d:\test\dblog
```

Read *Chapter 3, Configuration: SQL.INI and the Registry* for more for detailed information on all the configurable keywords.

## Creating a database

You must establish a server connection to create a database. You can do this with SQLTalk's SET SERVER command or the SQL/API's *sqlcsv* function.

There are a number of ways to create a database. Whatever interface you use, the result is the same: the database template file, *start.dbs*, is copied to a new *<database\_name>.dbs* file.

|                                 |                                                                                     |
|---------------------------------|-------------------------------------------------------------------------------------|
| SQLTalk                         | Use the CREATE DATABASE command to create a database and install it on the network. |
| SQL/API                         | Use the <i>sqlcre</i> function to create a database.                                |
| From an operating system prompt | Use an operating system command to manually copy <i>start.dbs</i> to a new file.    |

This example shows how to create a database using SQLTalk:

```
SET SERVER SERVERP/PASSWORD;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO

CREATE DATABASE EMP;
DATABASE CREATED
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO
 EMP

```

## Deleting a database

You must establish a server connection to delete a database. You can do this with SQLTalk's SET SERVER command or the SQL/API's *sqlcsv* function. There are a number of ways to delete a database.

|                                 |                                                                                                                                                                                                                                                               |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLTalk                         | Use the DROP DATABASE command to delete a database and its log files. SQLBase deletes both the database and log file directories (if they are different). This command also removes the <i>dbname</i> keyword from the server's configuration file (SQL.INI). |
| SQL/API                         | Use the <i>sqldel</i> function to delete a database.                                                                                                                                                                                                          |
| From an operating system prompt | Use an operating system delete command to manually delete a <i>.dbs</i> file.                                                                                                                                                                                 |

This example shows how to delete a database in SQLTalk:

```
SET SERVER SERVERP/PASSWORD;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO
 EMP

SHOW CONNECT;

Username: DBA
Database: DEMO
Cursor number: 1

DROP DATABASE EMP;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO

```

## Deinstalling a database

You must establish a server connection to deinstall a database. You can do this with SQLTalk's SET SERVER command or the SQL/API's *sqlcsv* function. You can take a

database offline to prevent users from logging on to it. This is useful when you want to restore and recover a database. There are a number of ways to deinstall a database.

|                                 |                                                                                                                                                                                                               |
|---------------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLTalk                         | Use the <code>DEINSTALL DATABASE</code> command to remove a database from the network.<br><br>This command also removes the <i>dbname</i> configuration entry from the server's configuration file (SQL.INI). |
| SQL/API                         | Use the <i>sqlded</i> function to deinstall a database.<br><br>This command also removes the <i>dbname</i> configuration entry from the server's configuration file (SQL.INI).                                |
| From an operating system prompt | Via an editor, remove the <i>dbname</i> configuration entry from the appropriate section of the server's configuration file (SQL.INI).                                                                        |

This example shows how to deinstall a database using SQLTalk:

```

SET SERVER SERVERP/PASSWORD;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO
 EMP

SHOW CONNECT;

Username: DBA
Database: DEMO
Cursor number: 1

DEINSTALL DATABASE EMP;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO

```

## Re-installing a database

You must establish a server connection to re-install a database. You can do this with SQLTalk's SET SERVER command or the SQL/API's *sqlcsv* function. There are a number of ways to re-install a database.

|                                 |                                                                                                                                                                             |
|---------------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SQLTalk                         | Use the INSTALL DATABASE command to re-install a database on the network.<br><br>This command also adds a <i>dbname</i> entry to the server's configuration file (SQL.INI). |
| SQL/API                         | Use the <i>sqlind</i> function to re-install a database.<br><br>This command also adds a <i>dbname</i> entry to the server's configuration file (SQL.INI).                  |
| From an operating system prompt | Via an editor, add a <i>dbname</i> configuration entry to the appropriate section of the server's configuration file (SQL.INI).                                             |

This example shows how to re-install a database using SQLTalk:

```
SET SERVER SERVERP/PASSWORD;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO

SHOW CONNECT;

Username: DBA
Database: DEMO
Cursor number: 1

INSTALL DATABASE EMP;
SHOW DATABASES ON SERVER SERVERP;

DATABASES ON SERVER: SERVERP
 DEMO
 EMP
```

## Loading and unloading a database

Unloading data involves “dumping” database information and Data Definition Language (DDL) commands to a flat file. You can then load this information to a new

database. You can also use the load operation to load information to SQLBase from an external flat file.

SQLBase uses the server to perform load and unload processes. This relieves the client of unnecessary processing, reduces network traffic, and allows the server to load and unload the data locally on its own machine. It also means that you can use the SQL/API *sqlrlo* and *sqlwlo* functions to read and write data, a buffer at a time.

Within a stored command or procedure, you can load and unload to the server but *not* to the client.

SQLBase load and unload operations handle large amounts of data, especially for large tables or databases. Load/unload files consume storage space and can be expensive to transmit. There is a compression option to reduce the size of these external files. You can compress the external file when you unload it, and decompress it when you reload it. Compression occurs at the server before transmission over the network.

## Loading

The following are methods to load database information:

|         |                                                                               |
|---------|-------------------------------------------------------------------------------|
| SQLTalk | Use the LOAD command to load database information from a flat file.           |
| SQL/API | Use the <i>sqlldb</i> function to load database information from a flat file. |

A load operation can restore data from a backup file created with a SQLBase unload operation, or can enter data into the database from an external file. The external file can be in SQL, ASCII or DIF format. You can create the file either manually or with the SQLBase unload methods listed in the next section. Do not edit these files manually. If you try to add commands such as COMMIT or ROLLBACK to the file, the load will fail.

Note that during the load operation, if ALTER TRIGGER commands are encountered, they are automatically processed.

## Unloading

The following are methods to unload database information to a flat file:

|         |                                                                               |
|---------|-------------------------------------------------------------------------------|
| SQLTalk | Use the UNLOAD command to unload database information to a flat file.         |
| SQL/API | Use the <i>sqlunl</i> function to unload database information to a flat file. |

A SQLBase unload operation allows you to back up a database or transfer data from a database to another program through interchange formats. An unload does not unload invalid stored commands.

When triggers are encountered during an unload operation, the status of the trigger is checked. If the trigger is disabled, the unload operation generates an `ALTER TRIGGER triggername DISABLE` statement for that trigger, which immediately follows the trigger's `CREATE TRIGGER` statement. If the trigger is enabled, the unload operation takes no action.

## Segmenting a LOAD/UNLOAD file

You can split a load/unload external file into segments spanning multiple disks. Segmenting an external file allows you to unload and load databases that might exceed single disk or system unit limits. For example, you may need to unload a database that exceeds 2 gigabytes into multiple file segments. Segmenting external files also lets you take advantage of available space that is spread out over several disks.

When unloading to a segmented external file, you designate how many bytes of information should reside in each file segment, and where each segment will reside. SQLBase unloads the database information sequentially to the file segments, putting the specified amount of information in each file segment. You can create a segmented file for any type of UNLOAD.

### Control file

To manage the file segments, SQLBase uses a *control file*. This file contains the following information:

- A prefix for the file segment names
- The destination directories where the file segments reside
- The maximum size of each file segment (for an unload operation only)

There are two types of control files for segmenting a LOAD/UNLOAD file:

- unload
- load

**Unload control file.** You must create the unload control file yourself using the instructions in the following section, *Creating the control file*. When running the unload operation, you then specify this control file name with the `CONTROL` clause. SQLBase automatically generates and unloads the database information sequentially to the file segments named in the control file.

Create the control file at the same location where you are going to store the external files. You can create the unload control file on either the client or server machines.

When you run UNLOAD, use the ON SERVER or ON CLIENT clause to tell SQLBase where the file is located. When it unloads information, SQLBase generate the unload file segments on the same machine as the unload control file, including any network drives.

If you do not supply a specific path when you run the unload operation, SQLBase assumes that the unload control file resides in the C:\Program Files\Gupta directory on the machine (or connected network drive) as specified by the ON SERVER or ON CLIENT clause.

**Load control file.** When you run an unload command with a control file, SQLBase automatically generates a corresponding *load control file* as output, as well as the unload file segments. When you subsequently run the load operation, use this load control file name.

You should use the SQLBase-generated load control file whenever possible, without making changes to it. However, certain situations may require a new or modified load control file, such as loading information that was unloaded from a non-SQLBase database, or moving the file segments since the UNLOAD. To create or modify a load control file, use the guidelines shown in the next section, *Creating the control file*.

SQLBase puts the new load control file in the same directory as the unload control file you created. If you do not supply a specific path, SQLBase assumes that the load control file resides in the C:\Program Files\Gupta directory.

The file segments must either reside on or be accessible from the same machine as the load control file.

## Creating the control file

To create a control file, use any editor on your system. The file must be in ASCII format, and contain the following information:

```
FILEPREFIX <filename prefix>
DIR <destination dir>SIZE <maximum size of the unload
 segment file in megabytes>
DIR <destination dir>SIZE <maximum size of the unload
 segment file in megabytes>
...
```

---

**Note:** The SIZE parameter only appears in the unload control file, not the load.

---

The following subsections describe these fields.

- FILEPREFIX

This parameter names the unload file segments. SQLBase names each unload file segment with this prefix, and appends an ascending value *n*. It also uses this file prefix to name the load control file, appending an extension of *.lcf*.

Example:

If you specify a file prefix *dfs*, SQLBase generates the following file segments during the UNLOAD operation:

```
dfs . 1
dfs . 2
dfs . 3
...
```

SQLBase also creates a load control file called *dfs.lcf*

The number of segments you can create is limited only by your system; SQLBase itself does not set a limit.

---

**Note:** Make certain that there are no existing files in the unload control file directory with the same name as the eventual load control file. SQLBase will overwrite any existing files that have the same name as the load control file.

---

- **DIR**

Use this parameter to name the destination directory (Windows Servers) or volume name (NetWare Server) for the file segments. You can put all the file segments in one directory, or split them out to different directories. SQLBase unloads the information to the file segments according to their listed order in the control file.

Each line specifies one file segment.

The directories can also include network drives, as long as the destination machine can access these drives.

- **SIZE**

This parameter specifies the maximum integer size (in megabytes) allowed for the specified file segment. During an unload operation, SQLBase dumps this exact amount of data to that file segment.

This parameter only appears in the unload control file, not the load.

Valid values are null or integer values of 1 through 2048 (in megabytes). Use the following calculation to determine the maximum number of bytes you can allocate:

$$\text{bytes} = (\text{size} * 1048576)$$

which is the maximum file size common across most systems.

If you wish to limit the size of your files, you can specify multiple file segments of a certain size, like the following Windows NT example:

Example:

```
DIR c:\unldir\ SIZE 50
DIR c:\unldir\ SIZE 50
DIR c:\unldir\ SIZE 50
```

With this control file, SQLBase unloads 50 megabytes of information to each of the three file segments.

The sum of all the sizes must be large enough to write all the unload information. If you do not know how much space you need, you can specify an approximate number of segments and their size. Depending on the amount of data you have, the last file segment SQLBase dumps to may or may not reach the SIZE amount. Any leftover segments are ignored; they do not tie up the space you designated for them.

---

**Note:** External files are not pre-allocated. If you direct SQLBase to create external files on the server machine, remember to consider the growth of log, history, and temporary files when you estimate the necessary available free space.

---

Note that since SQLBase dumps exactly the amount of data you specify, database objects and associated data may be split across multiple files.

If you build your own load control file, SQLBase ignores any SIZE parameters you create.

## Control file example

The following Windows NT example shows an unload control file:

```
FILEPREFIX dbs
DIR c:\unldir\ SIZE 100
DIR d:\unldir\ SIZE 50
DIR e:\unldir\ SIZE 200
```

Note that while SQLBase ignores any white space in this file, you cannot “wrap” a command statement to continue on the next line; SQLBase interprets new line characters as the end of an entry.

During an unload operation, this control file tells SQLBase to unload database information in the following order:

1. 100 megabytes of information to a file called **c:\unldir\dbs.1**
2. 50 megabytes of information to a file called **d:\unldir\dbs.2**
3. 200 megabytes to a file called **e:\unldir\dbs.3**

SQLBase also generates a corresponding load control file called *dbs.lcf* in the same directory as the unload control file. The following Windows NT example shows this file:

```
FILEPREFIX dbs
DIR c:\unldir\
DIR d:\unldir\
DIR e:\unldir\
```

SQLBase loads the information in the order listed. Note that there is no **SIZE** parameter.

## Error logging and recovery

Both the SQLBase loading and unloading commands can create a message log file. These message log files are useful for tracking activities and errors occurring during the load and unload processes.

For SQL or ASCII (not DIF) format, the **LOAD** command also has a option to restart a load operation from a particular line number in the load file. If you fix a data or statement error encountered during a load operation, use this clause to restart the load at the point where the error occurred, rather than having to restart the load from the beginning.

The line number for the **START AT** clause for a DDL statement must be the first line of the DDL command. For an **INSERT**, the line number for the **START AT** clause must be either the first line of the **INSERT** command or one of the line numbers that corresponds to a row of data you are inserting.

Note that with segmented loads, the line numbers are cumulative in the load file segments. To restart a segmented load from a specific line number, you must determine yourself in which file segment the specified line number is located.

---

**Note:** To avoid possible loss of data, issue a **COMMIT** statement immediately after a data or statement load error unless you are running SQLTalk in BAT mode, which performs an implicit **COMMIT** upon exiting.

Network or server failures are generally not recoverable. The existing loaded data is rolled back; you cannot restart the load from a specific line in this situation.

---

## Load performance enhancements

To improve load performance and preserve space, issue the following commands when executing a load:

```
set recovery off;
lock database;
```

```
load [sql | ascii | diff] external_file on server;
commit;
unlock database;
set recovery on;
```

## Creating a read-only database

SQLBase allows you to create read-only databases. This is useful for CD-ROM drives, as well as for other purposes:

- You can ensure that a database is altered only during certain hours of the day.
- You can have one central updatable database and distribute read-only copies of it to other sites.

To establish a database as read-only, use SQLTalk's SET command:

```
SET READONLYDATABASE ON;
```

You can turn READONLYDATABASE on or off from a C program using the *sqlset* function with the SQLPROD parameter.

You must be the only user connected to the database to turn READONLYDATABASE on or off. The default setting is off.

To display the setting of READONLYDATABASE, use SQLTalk's SHOW command:

```
SHOW READONLYDATABASE;
```

Read-Only isolation level is disabled when READONLYDATABASE is on. You cannot set the isolation level of your transaction to Read-Only mode.

Transaction log files are disabled once READONLYDATABASE has been turned on.

To turn off read-only mode, enter the following command:

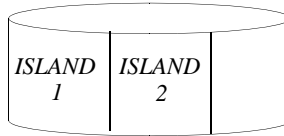
```
SET READONLYDATABASE OFF;
```

You must also set the configuration file's *tempdir* keyword or your operating system's file *tempdir* environment variable to tell SQLBase where to store temporary files. SQLBase actually creates a subdirectory in the directory pointed to by *tempdir* with the same name as the database. It is here that it stores the temporary files.

# Partitioning a database

This section covers setting up a database on partitioned volumes. It also introduces commands to maintain a partitioned database.

## Step 1: Create a database area



VOL1



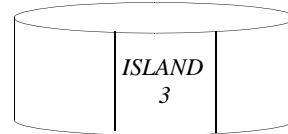
VOL2

```
CREATE DBAREA ISLAND1 AS VOL1:\GUPTA\ISLAND1 SIZE 5;
CREATE DBAREA ISLAND2 AS VOL1:\GUPTA\ISLAND2 SIZE 10;
CREATE DBAREA ISLAND3 AS VOL2:\GUPTA\ISLAND3 SIZE 10;
```

## Step 2: Create a storage group



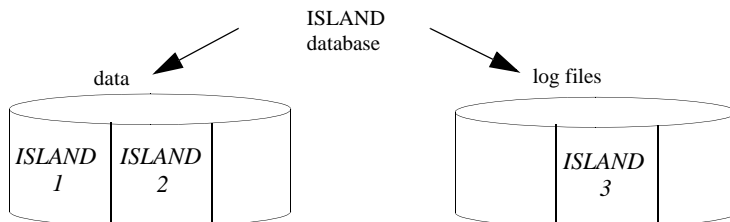
Storage Group ISLANDFILES



Storage Group ISLANDLOG

```
CREATE STOGROUP ISLANDFILES USING ISLAND1, ISLAND2;
CREATE STOGROUP ISLANDLOG USING ISLAND3;
```

## Step 3: Create a database



Storage Group ISLANDFILES

Storage Group ISLANDLOG

```
CREATE DATABASE ISLAND IN ISLANDFILES LOG TO ISLANDLOG;
```

*Steps for partitioning a database*

## Step 1: creating a database area

The first step in partitioning a database is to create a database area of a specified size either within the file system or in a raw partition. This process allocates room in which to create storage groups for your databases.

For example, to create the database area ISLAND1 in the \Gupta directory in volume VOL1 with a size of 5 megabytes, specify:

```
CREATE DBAREA ISLAND1 AS VOL1:\Gupta\ISLAND1 SIZE 5;
```

Specify a name for the database area with no more than 18 characters and specify the size of a database area in megabytes. Specifying a database area size is mandatory. The maximum size is limited by available disk space.

SQLBase returns an error if the file already exists or the disk space is already being used for another database area. It also returns an error if a file of the specified size cannot be created or if the actual size of the partition is smaller than the specified size of the area.

## Step 2: creating a storage group

The next step is to create a storage group (STOGROUP) by giving a name to a list of database areas. For example, to create the storage group ISLANDFILES for database areas ISLAND1 and ISLAND2, specify:

```
CREATE STOGROUP ISLANDFILES USING ISLAND1, ISLAND2;
```

Specify a name for the storage group with no more than 18 characters.

In the Novell environment, if the NetWare volumes containing the database areas are not mounted, SQLBase returns an error when you try to create a database.

## Step 3: creating a database

The last step is to create a new database in a storage group. You can also select a separate storage group for the log files.

For example, to create the database ISLAND in the storage group ISLANDFILES, and place the log files in a separate storage group called ISLANDLOGS, specify:

```
CREATE DATABASE ISLAND IN ISLANDFILES LOG TO ISLANDLOG;
```

Specify a name for the database with no more than eight characters.

If you specify a database storage group but do not specify a storage group for the logs, log file space is allocated in the database storage group. If you do not specify a storage group at all, the default storage group in the MAIN database is used, if one exists. If the MAIN database does not exist or you do not specify a default storage group, SQLBase creates the database and places it in the normal file system.

A database and its logs should not share any disks.

You should spread a database across as many disks as possible.

## Maintaining a partitioned database

This section steps you through tasks related to database areas and storage groups as well as extents.

### Database areas

#### Changing the size of a database area

You can increase or decrease the size of a database area.

For example, to increase the size of the database area ISLAND1 to 6 megabytes, specify:

```
ALTER DBAREA ISLAND1 SIZE 6;
```

You can decrease the size of a database area only if the space being deleted is not in use. SQLBase returns an error message if the file cannot be re-sized to the specified size.

#### Adding or dropping an area from a storage group

You can add or drop a database area to or from a storage group.

For example, to add the database area ISLAND3 to the storage group ISLANDFILES, specify:

```
ALTER STOGROUP ISLANDFILES ADD ISLAND3;
```

To then drop the database area ISLAND3 from the storage group ISLANDFILES, specify:

```
ALTER STOGROUP ISLANDFILES DROP ISLAND3;
```

These changes do not affect the storage of existing databases in the storage groups, but they do affect the future allocation of space for databases or log files which use the STOGROUP.

#### Deleting a database area

You can delete a database area as long as there are no database or log files using space within the area, and/or it is being used by a storage group.

For example, to delete the database area ISLAND1, specify:

```
DROP DBAREA ISLAND1;
```

## Storage groups

### Setting a default storage group

Once you create a storage group, all subsequent attempts to create a database result in SQLBase creating a partitioned database.

You can designate one of the existing storage groups as the default storage group.

For example, to set the default storage group to ISLANDFILES, specify:

```
SET DEFAULT STOGROUP ISLANDFILES;
```

The above example assumes that you already created the database area ISLAND1 and the storage group ISLANDFILES.

If you omit the storage group name, the default is set back to null. This signals SQLBase to create databases in the normal (non-partitioned) file system.

### Changing the database storage group

You can change the storage group for a database or its logs. For example, to change the storage group for ISLAND to ISLANDLISTER, specify:

```
ALTER DATABASE ISLAND STOGROUP ISLANDLISTER;
```

Existing database or log file space is not moved or affected by this command. The database you alter must already exist, or SQLBase returns an error message.

### Deleting a storage group

You can also delete a storage group if it is not in use by any database and it is not currently the default storage group.

For example, to delete the storage group ISLANDFILES, specify:

```
DROP STOGROUP ISLANDFILES;
```

## Extents

### Setting the extension size

SQLBase databases grow dynamically as data is added and expand in units called extents. When a database file becomes full, SQLBase must add another extent to the database. You are responsible for determining the size of the extent.

SQLTalk's SET EXTENSION command sets the extension size for both partitioned and non-partitioned databases. This is the syntax of the command:

```
SET EXTENSION <#kbytes>;
```

For partitioned databases, the value is rounded up to a 1 megabyte multiple.

## Free space in partitioned databases

To determine how much free space is left in a partitioned database, use the following query:

```
SET LINEWRAP ON;
column 1 width 8 heading 'Database';
column 2 width 8 heading 'stoareas';
column 3 width 8 heading 'areaname';
column 4 width 8 heading 'pathname';
column 5 width 8 heading 'areasize';
column 6 width 8 heading 'free%';
break on 1 2;
select a.name, b.stogroup, b.areaname, @trim(c.pathname),
 c.areasize, @nullvalue((100*d.extsize/c.areasize),0)
from syssql.databases a,
 syssql.stoareas b,
 syssql.areas c,
 syssql.freeexts d
where
 b.areaname = c.name
and c.name = d.name (+)
and ((a.stogroup = b.stogroup)
or (a.logstogroup=b.stogroup))
order by 1,2;
```

## Reorganizing a database

There are two types of fragmentation that can affect performance:

- Fragmentation of the database (\*.dbs) file.  
This happens when the database file does not have contiguous disk space.
- Fragmentation of tables within a database file.  
The tables in a database file can become fragmented over time by modifications to the data.

You can use a disk fragmentation utility such as Norton Utilities' Speed Disk (*sd*) to defragment a database file. Be sure to back up your database file first and check the database after you run the utility.

You can use SQLTalk's REORGANIZE command to defragment tables in the database to which you are currently connected. The REORGANIZE command:

- UNLOADs the database into a file
- Initializes the database
- Re-LOADs the database

SQLTalk uses a temporary file called *sqltmp.nnn* for unloading.

---

**Note:** Make a copy of the database file *before* executing this command. If an error occurs during the reorganization, both the temporary file and the database itself could be lost.

---

If you want to perform a complete defragmentation of both the database file and the tables in the database, first execute *sd* or a similar utility, and then execute SQLTalk's REORGANIZE command. For example:

```
REORGANIZE ;
UNLOAD COMPLETED
INITIALIZING DATABASE
STARTING TO LOAD

CREATE TABLE DBA.EMP (I INTEGER)
PCTFREE 10
TABLE CREATED

INSERT INTO DBA.EMP VALUES(:1)
LOADING TABLE DBA.EMP

PROCESSING DATA
50 ROWS LOADED
50 ROWS INSERTED
LOAD COMPLETED
```

For information on using the REORGANIZE command with encrypted databases, read *REORGANIZE with encrypted databases* on page 7-22.

## Checking database integrity

The SQL CHECK command performs integrity checks on an entire database or only specified portions of a database. Integrity checking consists of:

- Checking the integrity of the system and group free space data structures.
- Checking the system data and allocation structures.
- Verifying the table row count against the actual number of rows.
- Cross-checking each index against its base table.
- Checking the integrity of each row and index page.
- Ensuring that each page is part of an allocated structure or is on a free page list.

SQLBase reads every page of the database during an integrity check, resulting in the placement of a shared lock on each page.

| Command                     | Description                                                                                                                                     |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------|
| CHECK DATABASE              | This command performs an integrity check on the entire database.                                                                                |
| CHECK DATABASE SYSTEM ONLY  | This command verifies only system-defined tables and indexes; it ignores user-created tables and indexes.                                       |
| CHECK INDEX                 | This command performs an integrity check on only the specified index.                                                                           |
| CHECK TABLE                 | This command performs an integrity check on only the specified table and its indexes.                                                           |
| CHECK TABLE WITHOUT INDEXES | This command performs an integrity check on only the specified table and prevents SQLBase from verifying any indexes associated with the table. |

If SQLBase finds an integrity violation, it stops the integrity check and reports an error to the user. If the problem occurred on a user-defined object, SQLBase identifies the name of the object. If the problem occurred on a system-defined object, SQLBase provides either a description of the object or the name of the object.

If SQLBase discovers an object with integrity problems, you should drop it. If SQLBase discovers more extensive integrity problems, you may decide to restore a backed-up copy of the database. For example:

```
CHECK DATABASE ;
CHECK DATABASE SYSTEM ONLY ;
CHECK INDEX idx1 ;
CHECK TABLE emp ;
CHECK TABLE emp WITHOUT INDEXES ;
```

## Running multiple versions of SQLBase

### Version 8.5

Beginning with version 8.5, SQLBase makes it easy to run two or more database servers on a single computer. This is accomplished by:

- Prompting for a unique server name during installation, then using that name as an identifier in the configuration files, the Windows Services display, and SQLBase Management Console.

- Allowing configuration files to have any name and path, and allowing server and client applications to load a particular configuration file upon startup.
- Redesign of the Windows Registry structure related to SQLBase, removing the limitation of a maximum of two sets of installation information.

The main tool for managing multiple installations of SQLBase is the SQLBase Management Console. Refer to the online help in that tool for details on how to rename servers, change their properties, register new installations, and select a particular configuration file.

The Connectivity Administrator tool has also changed. This tool is used to view and edit the keyword values described in Chapter 3. Now, rather than automatically selecting a copy of SQL.INI from a hard-coded search sequence, it allows you the opportunity to specify the name and location of the configuration file that you wish to work with.

As illustrated earlier in this chapter (See “Starting and stopping SQLBase” on page 2.) the server executable accepts a command-line argument to force the use of a particular configuration file. Client applications, such as SQLTalk, also use a command-line argument in the same way - refer to the manual for the particular client application you are interested in. And the SQLBase API also has functions for specifying a particular configuration file.

The primary limitation on running multiple version 8.5 servers is SQLBase Resource Manager. This is a COM+ application, and Windows permits only one uniquely named COM+ application to be registered at a time. So you can run many copies of version 8.5, but only the most recently registered one can use the SQLBase Resource Manager.

How many copies of version 8.5 servers can be running simultaneously? It's limited only by the physical resources of your computer. However, earlier versions still have the same constraints that they were designed with. So you could have several copies of 8.5 servers running on your machine, but only one earlier version could be running simultaneously with them. The rules for earlier versions are described in the next section.

## Versions prior to 8.5

Since version 8.5 can coexist with earlier versions, the rules for specifying active installations in earlier versions are described here. With earlier versions, you can run two different versions of SQLBase. SQLBase uses the registry to allow only one version to be active at a time.

When SQLBase Server starts, it checks the registry to obtain configuration information. To allow for two independent installations of SQLBase, there are two different registry entries:

```
HKEY_LOCAL_MACHINE\SOFTWARE\Gupta\SQLBase\1
HKEY_LOCAL_MACHINE\SOFTWARE\Gupta\SQLBase\2
```

At any given time, only one of these entries can be active, as indicated by the “Active” item in one of the entries set to 1. SQLBase first checks if the Active item in ...\\SQLBase\\1 is set to 1, and if not, it then checks the Active item in ...\\SQLBase\\2. SQLBase then use the configuration information for whichever server is active.

If neither entry is marked as Active, SQLBase will be unable to start, and the following occurs:

- If set to run as a service, this results in an error message from the Windows Service Manager.
- If running as an application program, it results in an error message from SQLBase saying that a configuration file, such as message.sql or SQL.INI, cannot be located.

Use SQLBase Management Console to switch between the two installations. In SQLBase Management Console, right-click on the SQLBase Server Installations node, click the Active Installation item in the resulting context menu, and choose the server name that you want to activate.

For more, read *SQLBase Management Console (SMC)* on page 5-21.

Gupta recommends that you use SQLBase Management Console to switch installations because SQLBase stores two types of configuration information in the registry:

- Primary configuration information that is always used whether running as a program or as a service. For more, read *The Registry* on page 3-54.
- Service configuration information used only when running as a service. For more, read *Service information in the Windows Registry* on page 12-4.

While you can edit the primary configuration information directly with the Registry editor, you cannot edit the service configuration information.

When you run SQLBase as a Windows service, the two sets of configuration information must be synchronized. Although you can make a server active using the Registry editor, Gupta does not recommend it because you also need to change the service configuration information which is not possible with the Registry editor. If you only change the primary configuration information, then the Windows Service Manager will run its executable instance of SQLBase using the primary configuration information for the other installation.



## Chapter 7

# Security and Authorization

---

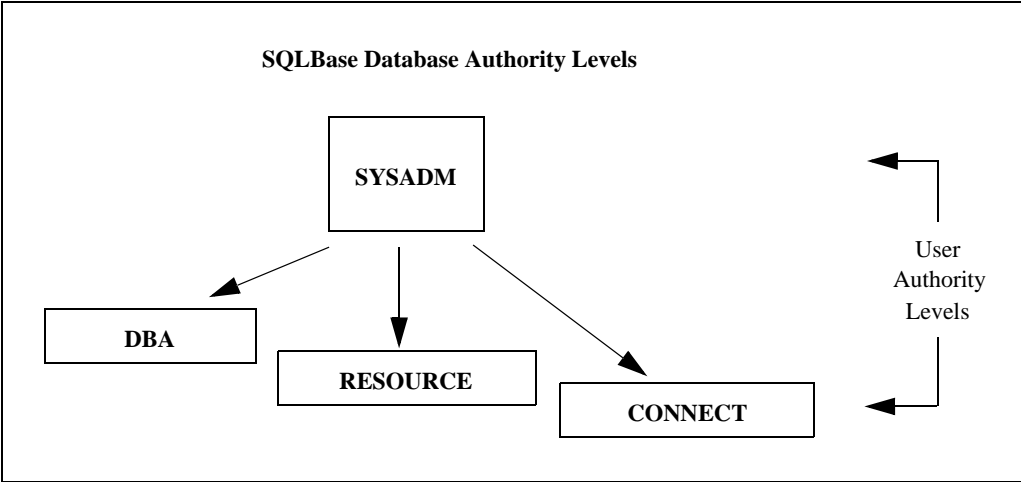
This chapter introduces you to SQLBase's security and authorization features.

- *Database authority* controls who can log on to SQLBase and the actions they can perform.
- *Table and view privileges* let users share some data while keeping other data private.
- *EXECUTE privileges* control who can access stored procedures.
- *Server security* controls who can perform server administrative operations.
- *Database page encryption* prevents unauthorized reading of data by viewing data files directly.
- *Data page alteration protection* prevents unauthorized changes to data by editing data files directly.
- *Transmission security* prevents unauthorized viewing of data directly as it is transmitted across a communication medium.
- *Delayed password validation* prevents unauthorized data connections made by password “guessing” programs.
- *Remote file protection* prevents SQL/API applications from reading and writing files on database servers

# Database authority

Database authority controls who can access a database and what they can do once connected to the database. SQLBase controls access with usernames and passwords.

SQLBase has 4 levels of users as shown in the following diagram.



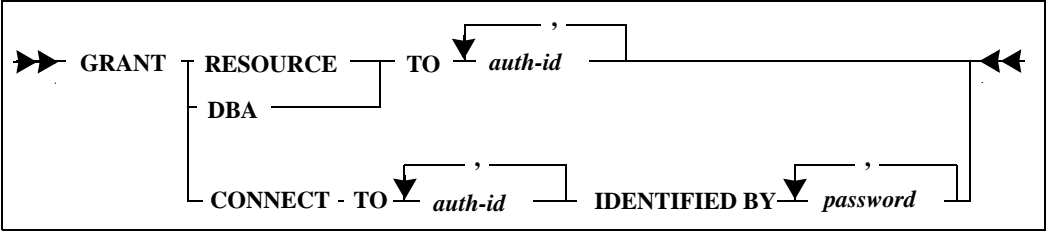
*Database authority levels*

The authority levels are hierarchical:

| Authority Level | Description                                                                                            |
|-----------------|--------------------------------------------------------------------------------------------------------|
| SYSADM          | Creates user accounts and designates users' authority levels and passwords.                            |
| DBA             | Grants, changes, or revokes the object privileges of any user.                                         |
| RESOURCE        | Creates and drops objects. Grants, changes, or revokes the privileges of other users on those objects. |
| CONNECT         | Accesses objects, but cannot create them.                                                              |

## Creating a user account

SYSADM alone has responsibility for establishing an account for each user by assigning each a username (auth-id), database authority level, and password. SYSADM does this using the GRANT command:



Only SYSADM can use this form of the GRANT command. Once you have created an account, you cannot change the username; you can only change the privileges.

SYSADM cannot grant other users SYSADM authority. The only thing that can be changed for SYSADM is the password.

The following authority levels can be granted by SYSADM:

| Authority Level | Description                                                                                                                                                                                                                                                                                                                                                                                                                                                     |
|-----------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONNECT         | This authority level must be granted before any other. This lets a user: <ul style="list-style-type: none"><li>• Logon to the database</li><li>• SELECT from other users' tables and views if the SELECT privilege has been granted to the user or to PUBLIC</li><li>• INSERT, UPDATE, and DELETE data into and from other users' tables if the appropriate privileges have been granted to the user or to PUBLIC</li><li>• Create views and synonyms</li></ul> |
| RESOURCE        | This authority level gives a user all CONNECT privileges and the right to: <ul style="list-style-type: none"><li>• Create tables and drop those same tables</li><li>• GRANT, change, or REVOKE privileges for those tables to and from other users</li></ul>                                                                                                                                                                                                    |
| DBA             | This authority level gives a user all CONNECT and RESOURCE privileges and all privileges on any object in the database as well as the right to: <ul style="list-style-type: none"><li>• GRANT, change, or REVOKE privileges for any object to and from other users</li></ul>                                                                                                                                                                                    |

Read the *SQLBase SQL Language Reference* for details about the GRANT command.

## SQLTalk examples

The following example establishes an account with CONNECT database authority, the username *user1*, and the password *secret*:

```
GRANT CONNECT TO USER1 IDENTIFIED BY SECRET;
CONNECT AUTHORITY GRANTED
```

The example below increases *user1*'s authority level to RESOURCE:

```
GRANT RESOURCE TO USER1;
RESOURCE AUTHORITY GRANTED
```

The example below increases *user1*'s authority level to DBA:

```
GRANT DBA TO USER1;
DBA AUTHORITY GRANTED
```

## Changing a user's password

There are two ways to change a user's password:

- SYSADM can change a user's password using the GRANT command and specifying the CONNECT option.
- A user can change his own password using the ALTER PASSWORD command.

➡➡ **ALTER PASSWORD** — *old password* — **TO** — *new password* —————<<⬅

Read the *SQLBase SQL Language Reference* for details about the ALTER PASSWORD command.

## SQLTalk examples

The following example shows SYSADM changing *user1*'s password from *secret* to *erehwon*:

```
GRANT CONNECT TO USER1 IDENTIFIED BY EREHWON;
CONNECT AUTHORITY GRANTED
```

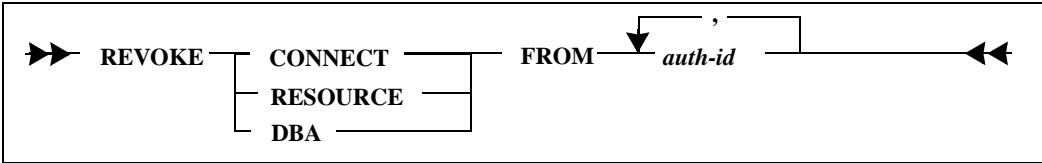
The example below shows USER1 changing the password from *erehwon* to *secret*:

```
ALTER PASSWORD EREHWON TO SECRET;
PASSWORD ALTERED
```

**Note:** Like all DDL operations in SQLBase, these operations can be rolled back and do not take effect for other users until committed. Also, note that uncommitted changes of this type cause locks to be held on the database.

## Revoking a user’s authority

The REVOKE command changes a user’s database authority level.



SYSADM can revoke these authority levels:

| Authority Level | Description                                                                                                                                                                                                                                              |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| DBA             | This reduces a user’s database authority to the CONNECT level: the user can no longer create or drop tables, or grant or revoke privileges on those table from other users. Any and all tables and views previously created by the user are kept intact. |
| RESOURCE        | This reduces a user’s database authority to the CONNECT level: the user can no longer create or drop tables, or grant or revoke privileges on those table from other users. Any and all tables and views previously created by the user are kept intact. |
| CONNECT         | This disallows the user access to the database. SYSADM must revoke privileges on tables and views <i>before</i> revoking CONNECT authority.                                                                                                              |

Read the *SQLBase SQL Language Reference* for details about the REVOKE command.

## SQLTalk examples

The following example takes away *user1*’s DBA authority:

```
REVOKE DBA FROM USER1;
DBA AUTHORITY REVOKED
```

The example below removes *user1*’s CONNECT authority:

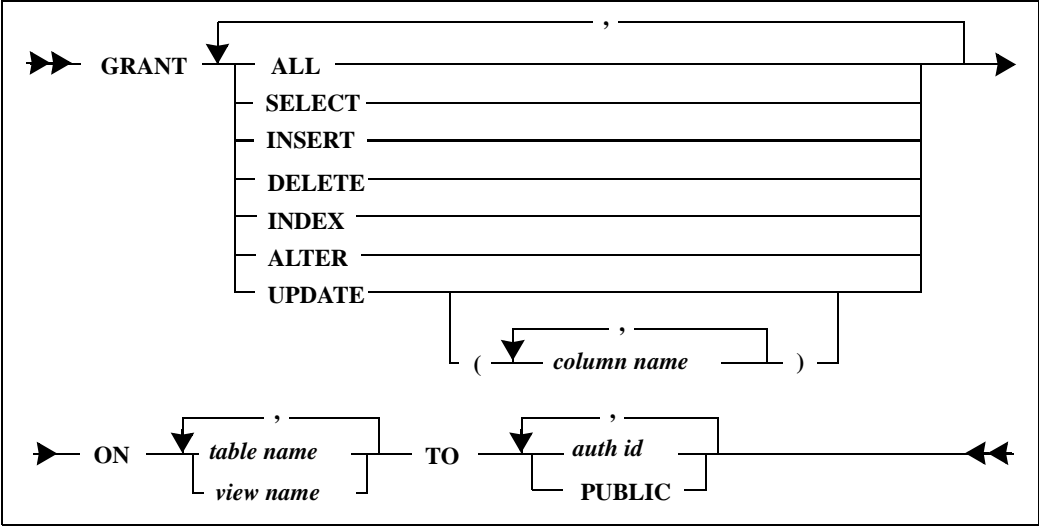
```
REVOKE CONNECT FROM USER1;
CONNECT AUTHORITY REVOKED
```

# Table and view privileges

A user with DBA authority can grant privileges on *any* tables or views in the database. A user with RESOURCE authority can grant privileges on only those tables created by him or on views based completely on tables created by him. Because a user with only CONNECT authority cannot create objects, he cannot grant privileges either.

## Granting privileges

A user who creates a table or view is the *owner* of that table or view and has all privileges on it. An owner can grant privileges on objects he owns to other users. He does this with the GRANT command.



GRANT INDEX and GRANT ALTER do not apply to views.

As an owner, you can grant the following privileges:

| Privileges | Description                                                        |
|------------|--------------------------------------------------------------------|
| SELECT     | Select data from a table or view.                                  |
| INSERT     | Insert rows into a table or view.                                  |
| DELETE     | Delete rows from a table or view.                                  |
| UPDATE     | Update a table and (optionally) update only the specified columns. |

| Privileges | Description                                         |
|------------|-----------------------------------------------------|
| INDEX      | Create or drop a table's indexes.                   |
| ALTER      | Alter a table.                                      |
| ALL        | Exercise all of the above table or view privileges. |

The keyword **PUBLIC** represents all users. By granting a privilege to **PUBLIC**, it means that all current and future users have the specified privilege on the table or view.

Read the *SQLBase SQL Language Reference* for details about the **REVOKE** command.

## SQLTalk examples

In the following example, *user1* gives *user2* the privilege to look at the data in *emp\_info*:

```
GRANT SELECT ON EMP_INFO TO USER2;
PRIVILEGE(S) GRANTED ON TABLE OR VIEW
```

The example below shows a DBA granting *user2* the privilege to look at the data in *user1*'s *emp\_info* table:

```
GRANT SELECT ON USER1.EMP_INFO TO USER2;
PRIVILEGE(S) GRANTED ON TABLE OR VIEW
```

The following example shows a DBA restricting *user2*'s **UPDATE** privilege on the *phoneno* column in *user1*'s *emp\_info* table:

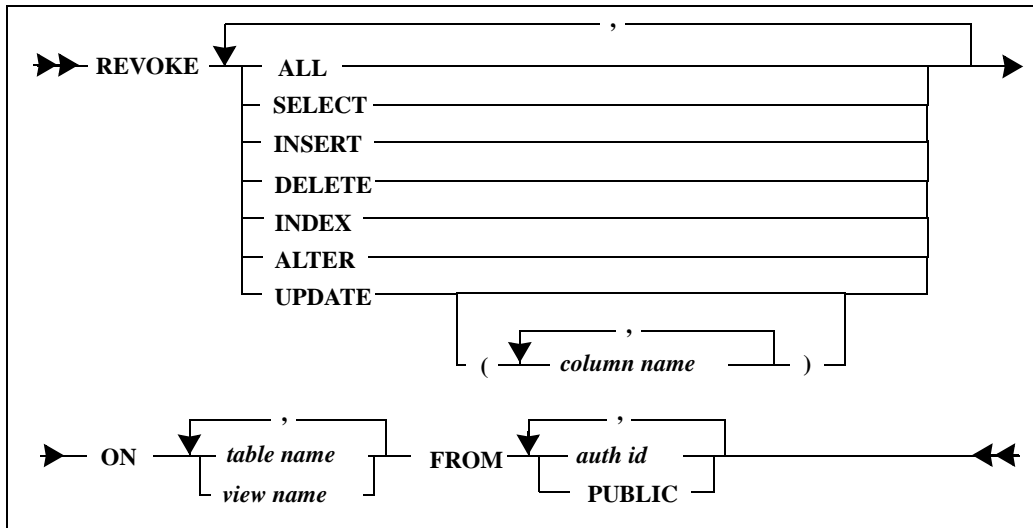
```
GRANT SELECT, UPDATE (PHONENO) ON USER1.EMP_INFO TO USER2;
PRIVILEGE(S) GRANTED ON TABLE OR VIEW
```

The next example shows a DBA giving all users the privilege of accessing the data in *user1.emp\_info*:

```
GRANT SELECT ON USER1.EMP_INFO TO PUBLIC;
PRIVILEGE(S) GRANTED ON TABLE OR VIEW
```

## Revoking privileges

The REVOKE command revokes privileges previously granted to a user.



Any user with the appropriate privileges for a table can revoke the privileges for the corresponding views.

The privileges are the same as for the GRANT command.

REVOKE INDEX and REVOKE ALTER do not apply to views.

The keyword PUBLIC represents all users. By revoking a privilege from PUBLIC, it means that all current users no longer have the specified privilege on the table or view.

Read the *SQLBase SQL Language Reference* for details about the REVOKE command.

### SQLTalk examples

The following example shows *user1* revoking *user2*'s privilege to access the data in *emp\_info*:

```
REVOKE SELECT ON EMP_INFO FROM USER2;
PRIVILEGE(S) REVOKED ON TABLE
```

The example below shows a DBA revoking *user2*'s privilege to access the data in *user1.emp\_info*:

```
REVOKE SELECT ON USER1.EMP_INFO FROM USER2;
PRIVILEGE(S) REVOKED ON TABLE
```

The following example shows a DBA revoking *user2*'s UPDATE privilege on the *phoneno* column.

```
REVOKE UPDATE (PHONENO) ON USER1.EMP_INFO FROM USER2;
PRIVILEGE(S) REVOKED ON TABLE
```

The example below shows a DBA revoking the public's privilege to the data in *user1.emp\_info*:

```
REVOKE SELECT ON USER1.EMP_INFO FROM PUBLIC;
PRIVILEGE(S) REVOKED ON TABLE
```

## Synonyms

When you access a table, view, or external function created by another user (once you have been granted the privilege), you must fully-qualify the object name by prefixing it with the owner's name:

authorization-id.object-name

For example:

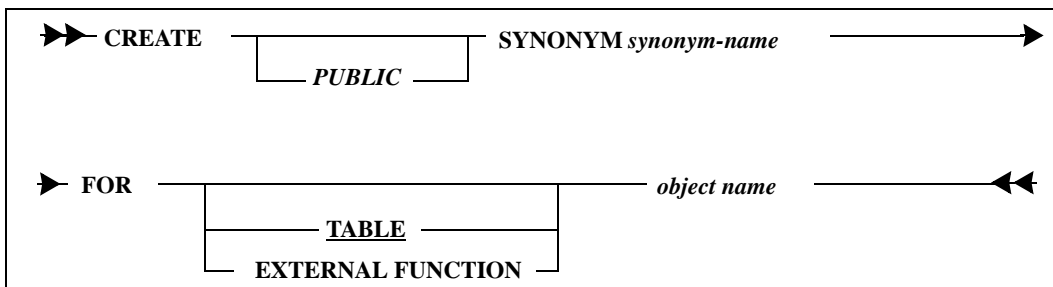
```
sysadm systables
user1.emp_info
```

If you try to access another user's table, view, or external function and you do not qualify the object name with the owner's name, SQLBase looks for a table, view, or external function owned by *you* with that name.

Synonyms save you typing by allowing you to refer to another user's table, view, or external function without having to fully qualify the name. A synonym is another name for a table, view, or external function.

## Creating a synonym

The CREATE SYNONYM command creates a synonym for the name of a table, view, or external function.



Synonyms for tables are stored in the SYSADM.SYSSYNONYMS system catalog table. Synonyms for external functions are stored in the SYSADM.SYSOBSYN system catalog table.

You must have at least one privilege on the table, view, or external function to create a synonym.

If many users need to access a table, view, or external function, the SYSADM, a DBA, or the owner can specify the PUBLIC keyword in the CREATE SYNONYM command to create a public synonym for all users with privileges on the table. This saves each user from having to create his own synonym. Be sure that only one table, view, or external function with that name exists in the database.

Note that if you want to create an external function synonym, you must provide the keyword EXTERNAL FUNCTION in the FOR clause. If no object type is specified in the FOR clause, the object type defaults to TABLE.

Read the *SQL Language Reference* for details about the CREATE SYNONYM command.

## SQLTalk examples

The following example shows a user creating a synonym for another user's (*user1*) table (*emp\_info*):

```
CREATE SYNONYM EMP FOR USER1.EMP_INFO;
SYNONYM CREATED
```

The example below shows the DBA creating a public synonym for all the users:

```
CREATE PUBLIC SYNONYM EMP FOR USER1.EMP_INFO;
SYNONYM CREATED
```

Note that in the previous examples, since no object type is included, the object type defaults to TABLE. These examples can also be presented as follows:

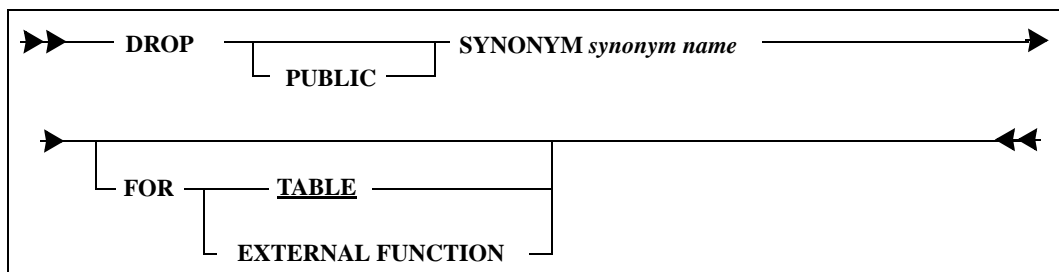
```
CREATE SYNONYM EMP FOR TABLE USER1.EMP_INFO;
CREATE PUBLIC SYNONYM EMP FOR TABLE USER1.EMP_INFO;
```

If you are a table owner and would like to make it convenient for other users to access a table without qualifying the name, then give a command like this:

```
CREATE PUBLIC SYNONYM EMP_INFO FOR EMP_INFO;
```

## Dropping a synonym

The DROP SYNONYM command deletes a specified synonym.



A synonym can only be dropped by its creator, SYSADM, or a DBA. SQLBase automatically drops any views based on the synonym as well.

Specify the PUBLIC keyword to remove a synonym that is available to all users.

Note that if you want to create an external function synonym, you must provide the keyword EXTERNAL FUNCTION in the FOR clause. If no object type is specified in the FOR clause, the object type defaults to TABLE.

Read the *SQLBase SQL Language Reference* for details about the DROP SYNONYM command.

## SQLTalk examples

The following example deletes a synonym:

```
DROP SYNONYM EMP;
SYNONYM DROPPED
```

The example below drops a PUBLIC synonym:

```
DROP PUBLIC SYNONYM EMP;
SYNONYM DROPPED
```

Note that in the previous examples, since no object type is included, the object type defaults to TABLE. These examples can also be presented as follows:

```
DROP SYNONYM EMP FOR TABLE;
DROP PUBLIC SYNONYM EMP FOR TABLE;
```

## Views

Granting privileges is one way of giving users access to database objects. Users can be granted select privileges without having privileges on the base table. The GRANT command gives you column level control by allowing you to specify whether users can see some or all of the columns in a table.

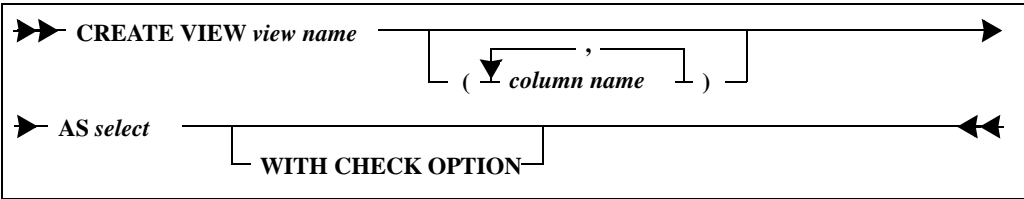
Views are another, more flexible way of giving users selective access to data. Views can enforce data security by allowing:

- Row-level access  
You can limit the rows that users can access by specifying a WHERE clause in the view definition.
- Column-level access  
Like the GRANT command, you can limit the columns that users can access by specifying only some of the table’s columns.

A user can create a view of a base table which only selects certain rows or columns and then grant privileges on that view.

### Creating a view

The CREATE VIEW command creates a view on one or more tables or views.



The creator of a view needs to have SELECT privileges on the columns of the base table.

The WITH CHECK OPTION causes all inserts and updates through the view to be checked against the view definition and rejected if the inserted or updated row does not conform to the view definition. If the clause is omitted, then no checking happens.

Read *SQLBase SQL Language Reference* for details about the CREATE VIEW command.

### SQLTalk examples

The following examples use this table:

| LNAME    | DEPT  | SALARY | PHONENO |
|----------|-------|--------|---------|
| =====    | ===== | =====  | =====   |
| Smith    | 355   | 32654  | 832     |
| Jones    | 150   | 28312  | 285     |
| Brown    | 700   | 48233  | 721     |
| Joyce    | 150   | 87935  | 145     |
| Chen     | 355   | 37666  | 222     |
| Herrlein | 700   | 18150  | 185     |

A DBA creates the view below so that *user1* can view the salaries and telephone numbers of employees in his department only:

```
CREATE VIEW EMP1
AS SELECT LNAME, SALARY, PHONENO
FROM EMP_INFO
WHERE DEPT = 355;
VIEW CREATED
```

Next, the DBA grants *user1* access to the view:

```
GRANT SELECT ON EMP1 TO USER1;
PRIVILEGE(S) GRANTED ON TABLE OR VIEW
```

Here is what the data looks like when *user1* selects from the view:

```
SELECT * FROM DBA.EMP1;

LNAME SALARY PHONENO
=====
Smith 32654 832
Chen 37666 222

2 ROWS SELECTED
```

The DBA creates the view below so that all employees can look at names and telephone numbers only:

```
CREATE VIEW PHONE
AS SELECT LNAME, PHONENO
FROM EMP_INFO;
VIEW CREATED
```

The DBA then makes the view publicly-accessible:

```
GRANT SELECT ON PHONE TO PUBLIC;
PRIVILEGE(S) GRANTED ON TABLE OR VIEW
```

Here is what the data looks like to anyone using the public view:

```
LNAME PHONENO
=====
Smith 832
Jones 285
Brown 721
Joyce 145
Chen 222
Herrlein 185
```

## Dropping a view

The DROP VIEW command deletes a specified view.

➡ **DROP VIEW** *view name* ⬅

A view can only be dropped by its creator, SYSADM, or a DBA.

SQLBase automatically drops privileges on the view, and drops any other view that depends partially or wholly on the specified view as well.

Read the *SQLBase SQL Language Reference* for details about the DROP VIEW command.

### SQLTalk examples

The examples below drop the views created previously:

```
DROP VIEW EMP1;
VIEW DROPPED

DROP VIEW PHONE;
VIEW DROPPED
```

## Execute privileges for stored procedures

To grant privileges for other users for stored procedures, use the SQL GRANT EXECUTE ON command. You can grant either your own privileges to other users, or grant them privileges of their own. To revoke privileges, use the REVOKE EXECUTE ON command. Read the *SQLBase SQL Language Reference* for information on these commands.

### External function privileges

When external functions are invoked within stored procedures, follow these rules for granting execute privileges:

- If a user has been granted execute with CREATOR privileges on a stored procedure, then the user does not need EXECUTE privileges on any external function invoked within the procedure. Only the CREATOR of the procedure needs to have EXECUTE privileges on the external functions.
- If a user has been granted EXECUTE with GRANTEE privileges on a stored procedure, the user must also have EXECUTE privileges on an external function invoked within the procedure.

In both cases, you use the `GRANT EXECUTE ON` command to grant privileges on external functions. This is the same command used for granting privileges on stored procedures.

When granting privileges for external functions with the `GRANT EXECUTE ON` command, note that the clauses `WITH CREATOR PRIVILEGES`, and `WITH GRANTEE PRIVILEGES` do not apply. These clauses only apply to stored procedures.

To revoke external function privileges, use the `REVOKE EXECUTE ON` command.

Read the *SQLBase SQL Language Reference* for information on these commands.

## System catalog

All authority levels and privileges are recorded in these SQLBase system catalog tables.

| System Table   | Description                                                                   |
|----------------|-------------------------------------------------------------------------------|
| SYSCOLAUTH     | Contains users' update privileges for individual columns of tables and views. |
| SYSTABAUTH     | Contains users' privileges for tables and views.                              |
| SYSUSERAUTH    | Contains the database authority level of each user.                           |
| SYSEXECUTEAUTH | Contains user's privileges for stored procedures.                             |
| SYSOBJAUTH     | Contains user's privileges for external functions.                            |

## Granting access to the system catalog

The system catalog tables are owned by `SYSADM`. This user can perform the following actions on the system catalog tables:

- Select data
- Create a view, index, or synonym
- Add and drop user-defined columns (with `ALTER TABLE`) as well as update them

However, no user (including `SYSADM`) can drop any of the original columns, nor insert or delete rows into or from the system catalog.

By default, `SELECT` privileges on the system catalog tables are granted to `PUBLIC` (all users). The exceptions are `SYSCOLAUTH`, `SYSCOMMANDS`, `SYSEVENTS`, `SYSOBJAUTH`, `SYSTABAUTH`, `SYSTRGCOLS`, `SYSTRIGGERS`, and `SYSUSERAUTH` which only `SYSADM` and a `DBA` can view. For sensitive data, this

may be undesirable. Users may not be able to tell what the actual data is, but they might be able to tell what *type of data* exists.

For this reason, SYSADM or a DBA can create selective views of the system catalog tables and grant users access only to the views. This gives users access to the system catalog data without letting them see everything. SYSADM or DBA should also revoke select access from PUBLIC unless it is needed.

The USER keyword is useful for creating views on system catalog tables because it represents the authorization-id of the current user. When creating a view on a system catalog table, SYSADM can specify USER in the WHERE clause of the SELECT command to restrict the data to only that appropriate for the current user.

For example, the following CREATE VIEW command retrieves data from SYSADM.SYSTABLES only for the tables that the current user owns:

```
CREATE VIEW MYTABLES AS
SELECT * FROM SYSTABLES
WHERE CREATOR = USER;
```

Read *Appendix A, System Catalog Tables* for descriptions of the SQLBase system catalog tables.

## Server security

At the server level, SQLBase has two types of security:

- Server connection passwords
- Server security passwords

### Server connection password

With server connection passwords, you set a password in *SQL.INI* and then specify the password when you connect to the server to perform administrative operations. This prevents unauthorized users from performing destructive operations on the server. For more, read *password* on page 3-42.

If you do not declare a server password in *SQL.INI*, any user can make a server connection. This option is not recommended for secure systems.

You can disable the use of a server connection password by setting the password to '\*' in *SQL.INI*. For more, read *Using only a server security password* on page 7-18.

Connecting to a server enables you to perform these operations:

- Backup, recovery, and restoration of a database and its logs
- Creation and deletion of a database and its logs
- Installation and deinstallation of a database

- Obtain details about server activity through the SQL/API

These operations are discussed in *Chapter 4, Databases* and profiled in *Chapter 6, DBA Operations*.

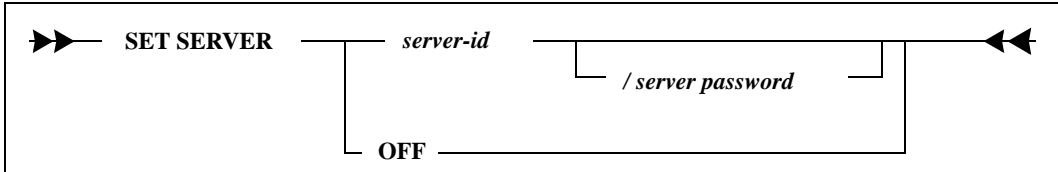
---

**Important:** You cannot make a server connection using ODBC or JDBC because these standards do not support the concept of a server-level connection.

---

## Establishing a server connection using SQLTalk

To establish a connection to a server, use SQLTalk's SET SERVER command.



When you are finished performing administrative operations, use the SET SERVER OFF command to break the server connection.

Read the *SQLTalk Command Reference Manual* for details about the SET SERVER command.

## Establishing a server connection using the SQL/API

Using the SQL/API, you make a connection by calling the *sqlcsv* function. In addition to the administrative operations listed above, using the SQL/API lets you abort a database or server process.

## Server security password

In addition to letting you perform all the administrative operations permitted by a server connection password, a server security password enables SQLBase to encrypt databases and detect unauthorized changes to database pages. Until you set the security password, SQLBase does not create encrypted databases or allow access to encrypted databases.

## Overview of secure servers

To set up a secure server, you start by connecting to the server and setting the server security password. You then connect to a database that you want to encrypt and give an ALTER DBSECURITY command where you specify a database encryption key. SQLBase then encrypts the database.

The next time you start the server, SQLBase automatically installs and allows access to unencrypted databases, but it does not install encrypted databases. SQLBase waits

until you connect to server and specify the server security password. SQLBase then installs encrypted databases and allows users to connect to them. After you disconnect, the encrypted databases continue to be available for users to connect.

## Setting a server security password

You can set the server security password interactively in these ways:

- If you are running Windows, select **Security, Set Security Password** in SQLBase server console.



- If running on a NetWare server, press **F4**.

Follow the same rules for creating a server security password that you follow for long identifiers such as column names (32 character maximum).

---

**Note:** By default, a new database does not have a database encryption key or a security encryption level and is compatible with non-secure versions of SQLBase. However, once you set a server security key for a database, it is not compatible with non-secure versions of SQLBase, even if the database security level is set to none.

---



---

**Warning:** If you specify a server security password longer than eight characters, you will not be able to connect to the server (with a SET SERVER command) with a pre-7.5 version of SQLTalk. Database connections are not affected however.

---

## Using only a server security password

If you want to have a completely secure server, then specify an asterisk for the password in *SQL.INI* and use only the server security password:

```
[dbntsrv]
servername=secure
password=*
```

This tells SQLBase to accept only the server security password for server connections, and removes any security risk of making the password visible in *SQL.INI*.

**Important:** Do not change the password in *SQL.INI* to asterisk until you have at least one encrypted database. Then, shut down SQLBase, change *SQL.INI*, and restart SQLBase. At that point you can use the server security password in the SQLTalk SET SERVER command or the sqlcsv function.

---

## Implications of using server security passwords

Setting the server security password enables SQLBase's security mode. This enables access to databases encrypted previously by this server set with the same security password.

SQLBase validates the given server security password against any currently encrypted databases, and if valid the databases are brought online. If no encrypted databases are present:

- and you are specifying the password through SQLBase Server Console, SQLBase assumes the given password the password is valid
- and you are specifying the password through the SQL/API, SQLBase assumes the given password is invalid

The password is used as part of the encryption process for any databases that are encrypted. Encrypted databases can only be accessed by a server set to the same security password in use at the time they were last accessed. SQLBase does not “remember” the security password once it shuts down, nor does it record the password anywhere.

---

**Warning:** If you forget the server password, it makes an encrypted database unusable and irrecoverable.

---

By design, there is no mechanism to recover data from an encrypted database if you lose the server password. Guessing at passwords is unlikely to help because SQLBase protects itself against this by becoming unresponsive to callers using invalid passwords. For more, read *Delayed password validation* on page 7-27.

## Specifying the server security password

Once you have set the server security password and enabled encryption for a database, SQLBase no longer automatically installs the database when it starts. SQLBase does not install the database until you specify the server security password. SQLBase will continue to bring unencrypted databases online automatically.

You can set the server security password locally and interactively in these ways:

- If you are running Windows, select **Security, Set Security Password** in SQLBase server console.

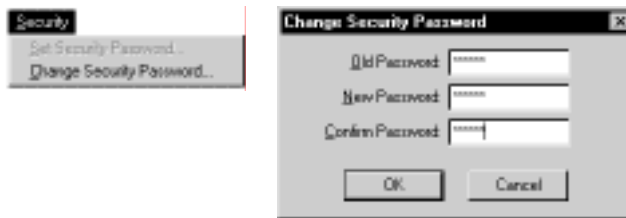
- If running on a NetWare server, press **F4**.

You can specify the server security password remotely in these ways:

- In SQLTalk, connect to a server using a **SET SERVER** command, specifying the server security password instead of the server connection password.
- In a program, connect to a server by calling the `sqlcsv` function and specifying the server security password instead of the server connection password.

## Changing the server security password

You change the server security password in SQLBase server console by selecting **Security, Change Security Password**. This item is grayed if no server security password has yet been set, either via the Security menu, or by a client performing a **SET SERVER** command or by the SQL/API `sqlcsv` function.



---

**Important:** You can only change the server security password locally on the server.

---

Selecting this menu option displays a dialog where you specify the old server password and the new server password. When you select this option, SQLBase checks to ensure that no users are connected to the server and users who try to connect while the option is selected are refused connection.

Changing the server password changes the signatures of the currently online databases to the new password. To back up or restore a database encrypted with a different server security password you must first give the **ALTER EXPORT** command. For more, read *ALTER EXPORTKEY* on page 7-24.

For more information, read *Backups and security configuration settings* on page 7-25.

## Database page encryption

The server security password allows access to encrypted databases via SQLBase. A database encryption key is used to encrypt a database to prevent other methods of accessing the stored data. Each database can have its own encryption key and security level.

---

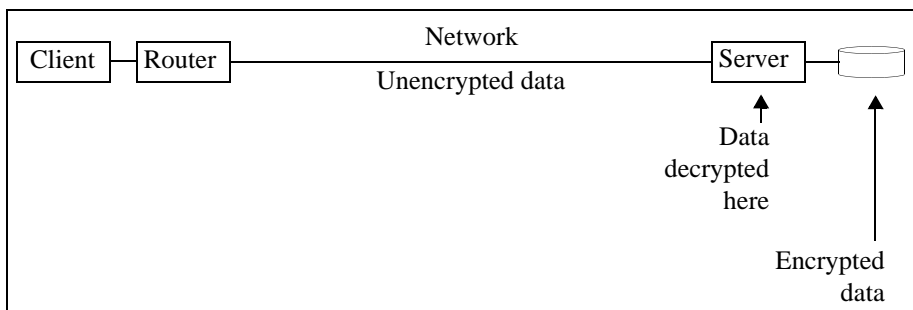
**Note:** Although it is possible to set a server security password in SQLBase Standard Edition, the other page encryption features discussed in this section are only available with SQLBase Treasury Edition 56 or Treasury Edition 128.

---

Database security levels refer to the type of encryption used to hide the data within the database. The choice of encryption is a trade off between performance and difficulty in breaking the data encryption.

There are four security levels: none, low, medium, and high. You set the security level for a database using the `ALTER DBSECURITY` command. For more, read *ALTER DBSECURITY* on page 7-24.

When you enable database page encryption, SQLBase stores database pages on disk in encrypted form. When users access the data, SQLBase decrypts the page and transmits it to the client unencrypted. In other words, it is the server that does the decrypting, not the client. Decryption is transparent to the client.



Note that you can encrypt the data as it is transmitted across the wire. For more, read *Transmission security* on page 7-25.

---

**Important:** The security levels are described generically here. The implementation details may depend on your locale. Check the release notes for more information.

---

## None

This level does not encrypt data.

## Low

In this security level, each database page is converted using a cryptogram (each character is replaced with a different character using a non-trivial formula). This security level discourages the casual observer using a file viewer such as a hex editor

but is not intended to be highly secure. However, it offers reasonably high-performance.

## Medium

This level is reasonably secure and performs better than the high level of security.

## High

This is the most secure, but poorest performing level. Depending on the version of SQLBase that you are using, this level may map down to medium security. Check the release notes for more information.

## Transaction log encryption

If you turn on any level of database page encryption other than none, then SQLBase encrypts transaction logs at the medium level.

## REORGANIZE with encrypted databases

When you perform a REORGANIZE, SQLBase does an unload (unencrypted) and then creates a new *unencrypted* database. Centura Software recommends that instead of performing a REORGANIZE with encrypted databases, you manually reorganize by following the steps below:

1. Perform a UNLOAD command.
2. DROP the database.
3. Create a new database.
4. Encrypt the new empty database
5. LOAD the database.
6. Delete the UNLOAD file

---

**Warning:** Unload files are unencrypted and if you intend to retain them you should take steps to protect them such as by using the encryption feature of a file compression utility.

---

## Database page alteration protection

SQLBase uses database page alteration protection to detect if data has changed in an unauthorized way.

There are three levels of database page alteration protection: none, 16-bit CRC, and SHA. You set the database page alteration protection level with the `ALTER DBSECURITY` command. For more, read *ALTER DBSECURITY* on page 7-24.

As with the differing levels of encryption, page protection is a trade off between efficiency and protection. The higher levels of encryption are very sensitive to page alterations, so protecting medium and higher encrypted database pages against alteration can be unnecessary if you assume that the encryption is sufficient protection for the data.

## None

With this level of protection, page data alteration is not tested directly by SQLBase. If the database was encrypted, however, it would be extremely unlikely that any part of a database page could be altered in anyway without completely corrupting the page, causing a database shutdown when SQLBase accesses the page.

## 16-bit CRC (cyclic redundancy check)

With this level of protection, SQLBase can detect most changes to the database. The CRC is stored in the page before any encryption.

If it detects a mismatch, SQLBase shuts down the database, reporting an (apparent) media failure.

## SHA (Secure Hash Algorithm)

This level of protection provides a digital signature for each database page. SHA keys are difficult to reverse and alteration is easily detected. This level provides substantially more protection than CRC. The digital signature is stored in the page before any encryption.

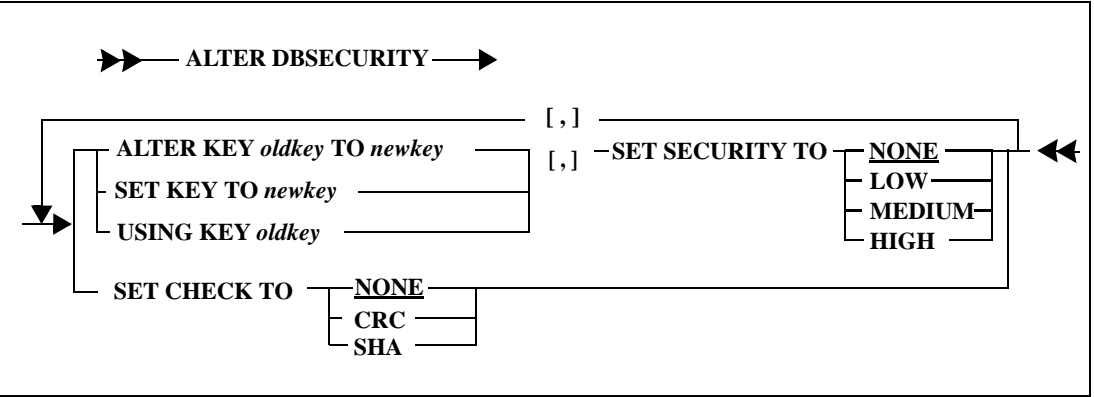
If SQLBase detects an SHA error, it shuts down the database, reporting an (apparent) media failure.

# ALTER DBSECURITY

You use the ALTER DBSECURITY command to change the database encryption key, the database encryption level, and the page alteration protection level.

**Note:** This command works only on SQLBase Treasury Edition 56 or Treasury Edition 128.

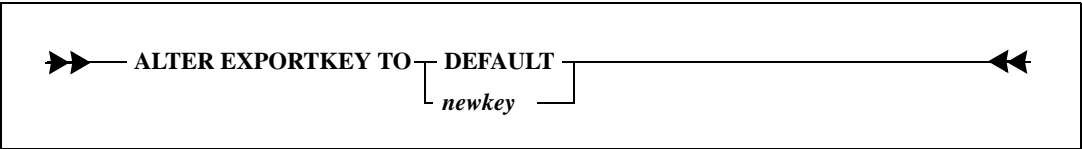
**Note:** You must be logged on as SYADM and you must be connected to a database to perform this command.



**Important:** Since this is a SQL command, you can compile and execute this command through the SQL/API.

# ALTER EXPORTKEY

You use the ALTER EXPORTKEY command to change the server security password used to encrypt databases. You use this command to export (BACKUP) and import (RESTORE) databases to and from another server with a different server security password.



During a restore, SQLBase tries to decrypt the database using the current server security password. If that password cannot decrypt the database, SQLBase uses the password specified by the most recent ALTER EXPORTKEY command. SQLBase then re-encrypts the database using the current server security password. During a backup, SQLBase uses the password specified by the most recent ALTER EXPORTKEY command to encrypt the database.

---

**Important:** Since this is a SQL command, you can compile and execute this command through the SQL/API.

---

## Backups and security configuration settings

You cannot rollforward past a change in the security configuration, including changing the server security password, the database page encryption, and database page alteration protection. If you try to do this, the restore stops just before the security change. After making a security change, you should restart your backup procedure by making a fresh backup of the database and its logs because previous versions of the database cannot be recovered beyond the security change. Therefore, do this after making a security change:

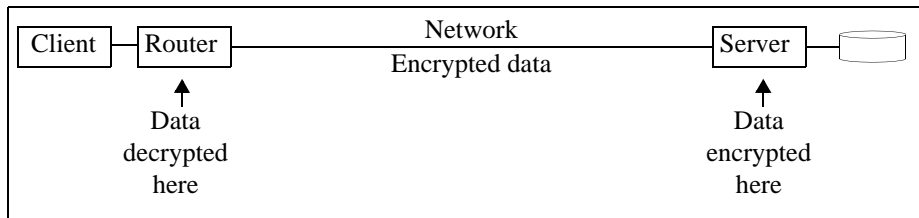
1. Shut down the server.
2. Perform an offline backup

OR

Start the server and perform an online backup.

## Transmission security

You can use transmission security between SQLBase clients and servers to prevent unauthorized users from seeing data as it is transmitted across the communications medium.



Transmission security is independent of database page encryption and database page alteration protection. Transmission can be used alone or in conjunction with the other two. You do not have to set a server security password (page 7-17) to use transmission security.

**Important:** You can use transmission security with ODBC, but not with JDBC.

You enable transmission security by specifying two keywords in *SQL.INI*: *secureapi* and *negotiateapi*.

secureapi keyword

Both the client and server use the *secureapi* keyword in *SQL.INI*:

- For the client, *secureapi* indicates the *actual* encryption used in messages between the client and server.
- For the server, *secureapi* indicates the *minimum* level of encryption of messages to which the server responds. The server always responds to the client using the same level of encryption that the client used, assuming that it accepts that level.

The table below shows the meaning of the settings of the *secureapi* keyword:

| Level  | Value | Client meaning        | Server meaning                                |
|--------|-------|-----------------------|-----------------------------------------------|
| None   | 0     | Use no encryption     | Accept any encryption level                   |
| Medium | 1     | Use medium encryption | Accept medium and high encryption levels only |
| High   | 2     | Use high encryption   | Accept high encryption level only             |

**Important:** The security levels are described generically here. The implementation details may depend on your locale. Check the release notes for more information.

If you do not set the *secureapi* value in *SQL.INI*, default values are used. For the client and server, this is 0 (none).

To achieve and maintain a connection, the level of encryption must be compatible. If a client requests an encrypted connection from a server that does not support encryption, the client receives an error.

For example, a client could have this in *SQL.INI*:

```
[win32client]
; always use high encryption
secureapi=2
```

The server could have this in *SQL.INI*:

```
[dbntsrv]
; accept non-encrypted messages
secureapi=0
```

## negotiateapi keyword

You specify this keyword on a client (in the [winclient] or [win32client] section) that has a router compliant with version 7.5 or later, to enable the client to connect to servers that do not support transmission security. For example, you can have a client that always tries to connect with some form of transmission security. However, if this client also has to connect to servers that do not support transmission security, this keyword lets the client make a connection without using transmission security.

To use this feature, set the secureapi keyword to 1 or 2 and set the negotiateapi keyword to 1. If the server does not support transmission security, it rejects the first attempt to connect and the router then tries to connect without transmission security which the server accepts.

An application can make connections to multiple servers, some using transmission security, some not.

---

**Note:** If a client uses high security and the server uses medium security, the connection always fails. In this situation, you must use medium security on the client to make the connection.

---

## Delayed password validation

SQLBase protects itself from password guessing programs using delayed password validation.

When SQLBase detects an invalid password or username, it delays responding to the caller. Valid connections are never delayed. “Password” refers to both database connections and server connections.

SQLBase increases the period of delay for every invalid attempt to connect, regardless of the source of the attempt. SQLBase only reduces the delay after a period of time where no invalid attempts to connect have occurred.

## Remote file protection

By default, a SQL/API application can call these functions to read and write files on a remote database server:

- sqlfgt
- sqlfpt
- sqlmop

- sqlmdl

To prevent any application from calling these functions, set the fileaccess keyword to zero (0) in the server section of SQL.INI.

## Tutorial

This section walks you through the steps for setting up a server security password, database encryption, database page alteration protection, and transmission security for a database server.

1. Start SQLBase.
2. In the SQLBase server console, choose menu items Security, Set Security Password.

---

**Important:** Be sure to choose a password that you will remember. The password you choose will be required to access all databases encrypted while the server has this password.

---



3. Start SQLTalk and make a server connection, specifying the server name and connection password (if any) as specified in *SQL.INI*:

```
set server server1/password;
```

You need to perform this command to do administrative operations.

4. Create a new database:

```
create database securedb;
```

5. Connect to the database:

```
connect securedb;
```

Be sure to connect as SYSADM (default) or you will not be able to perform the ALTER DBSECURITY command.

6. Using SQLTalk, set the database encryption key and the database page encryption level:

```
alter dbsecurity set key to key1 set security to high;
```

This process can take some time depending on the size of the database.

7. Exit SQLTalk and shut down SQLBase.
8. To set the server connection password to asterisk (\*), start the Connectivity Administrator from the Centura program group and click the **Server** tab:



9. Click the **Properties** button:



10. Click the **Optional** tab. Select the **Enable Server Connect Password** box (if not already selected), and in the **Server Connect Password** field, enter an asterisk.



After you change the password to asterisk, you use the server security password to connect to the server. This makes the server the most secure because the password is not recorded anywhere.

11. Click **OK** twice to exit Connectivity Administrator. A message displays stating that new configuration settings will not take effect until the next time SQLBase is started.
12. Start SQLBase and SQLTalk.
13. Check the Databases window in SQLBase server console. You will see that SQLBase did not bring the SECUREDB database online.
14. Connect to the server, specifying the server security password you selected in step 2 of this tutorial:

```
set server server1/password;
```

After you have encrypted a database, each time the server starts you must specify the server security password through SQLBase server console, SQLTalk, or the sqlcsv function to bring encrypted databases online. SQLBase will bring unencrypted database online automatically.

15. Check the Databases window again. You will see that securedb is now online.



16. Connect to the database:

```
connect securedb;
```

17. Set the database page alteration protection level:

```
alter dbsecurity using key key1 set check to sha;
```

This command may take some time depending on the size of the database.

18. To turn off database page encryption for a database, enter a command like this:

```
alter dbsecurity using key key1 set security to none;
```

Again this process can take some time depending on the size of the database.

19. To turn off database page alteration protection, enter a command like this:

```
alter dbsecurity using key key1 set check to none;
```

20. To backup a database that will be restored on a different server with a different server security password, first give an ALTER EXPORTKEY command, specifying the other server's password:

```
alter exportkey to password2;
```

Then back up the database:

```
backup snapshot to d:\backup;
```

21. To restore a database from a server with a different server security password, first give an ALTER EXPORTKEY command, specifying the other server's password:

```
alter exportkey to password2;
```

22. Then restore the database:

```
restore snapshot from d:\backup to securedb;
```

23. Exit SQLTalk and shut down SQLBase.

24. To enable transmission security on the server, start the Connectivity Administrator in the Centura program group.
25. Click the **Server** tab.
26. Click the **Properties** button.
27. Click the **Optional** tab. In the Encryption Level group, click **High Encryption Only**, and click **OK**.



28. To enable transmission security on the client, click the Connectivity tab, select SQLBase at the bottom of the list, and click the Properties button:



29. In the Encryption Level group, click High Encryption, click OK to close the Client Properties dialog, and then click OK to exit Connectivity Administrator.
30. Start SQLBase.
31. In SQLBase server console, set the server security password.
32. Start SQLTalk.
33. Connect to the securedb database. Your connection should be successful because the transmission security levels for the client and server are compatible.

34. Exit SQLTalk and shut down SQLBase.
35. Following the instructions for setting transmission security above, start Connectivity Administrator, set the client-side security level to No Encryption, and exit Connectivity Administrator.
36. Start SQLBase.
37. In SQLBase server console, set the server security password.
38. Start SQLTalk.
39. Try to connect to the securedb database. The connect fails and you get a message saying that the session is closed.
40. Exit SQLTalk and shut down SQLBase.
41. Start Connectivity Administrator, turn off both the client-side and the server-side transmission security level, and exit Connectivity Administrator.



## Chapter 8

# Backing Up and Restoring Databases

---

This chapter discusses SQLBase's backup and restore options. These topics are covered:

- Backup planning
- Making an online backup of a database
- Making an offline backup of a database
- Restoring an online backup of a database
- Restoring an offline backup of a database
- Backing up and restoring a partitioned database

If you are running a distributed transaction, there are special issues to resolve before restoring a database. Read *Chapter 9, Distributed Transactions* for more information.

## Making a backup plan

This section describes how to make a backup plan.

### Crash recovery

A database can be damaged in a number of ways such as by a power failure or an operator error in bringing down the server. When an event like this happens, SQLBase tries to restore the database to a consistent state by performing crash recovery automatically when a user attempts to connect. Crash recovery consists of using the transaction logs to redo any committed transactions which had not yet been written to the database and to undo any uncommitted transactions which were active when the server crashed.

There are situations where SQLBase will not be able to return a database to a consistent state, such as when the transaction logs have been damaged during a media failure, or if transaction logging has been disabled by the operator.

Use the *fail.sql* file to review information. This file contains information regarding failure and error situations that occur, generated by SQLBase, and should be reviewed any time an unusual event occurs with the database.

### Media recovery

Maintenance is a necessary part of a database administrator's job, and involves preparing for events such as a disk head crash, an operating system crash, or when a user inadvertently drops a database object. You can recover from media failures and user errors which have damaged a database *if* you make database and log file backups regularly. Making backups of your database and log files from which you can restore the database is the only way you can prevent loss of data.

*How often you backup the database and its log files depends on how much data you can afford to lose.* In general, the following are good guidelines:

- Backup the database once a week.
- Backup the transaction log files once a day.

You can minimize loss of data due to a media failure by backing up transaction logs frequently. You should backup all logs made after the last database backup so that they can be used to recover the database up to the point of the last log.

In addition, you should save the database and log files from the last several sets of backups taken. For example, if you make a BACKUP SNAPSHOT every Sunday, and make log backups every night, a backup set would consist of the snapshot, and Monday through Saturday's log file backups. You should save several of these weekly sets. Never rely on just one backup!

**Note:** Never delete transaction log files. SQLBase automatically deletes log files either when they are backed up, or when they are no longer needed for transaction rollback or crash recovery, depending on whether LOGBACKUP is on or off. A database file may be useless without its associated log files.

---

To ensure a relatively fail-safe strategy for backing up and restoring the database to log files, retain two cycles of backups as follows:

Cycle 1 Day 1 backup database and log files

Day 2 backup log files

...

Day n backup log files

Cycle 2 Day 1 backup database and log files

Day 2 backup log files

Cycle 3:... (Repeat process)

Only when Cycle 3 is successful, should cycle 1 be eliminated.

Also, note that Day n for the previous cycle should coincide with Day 1 of the next cycle to minimize loss of data.

## Backing up a database

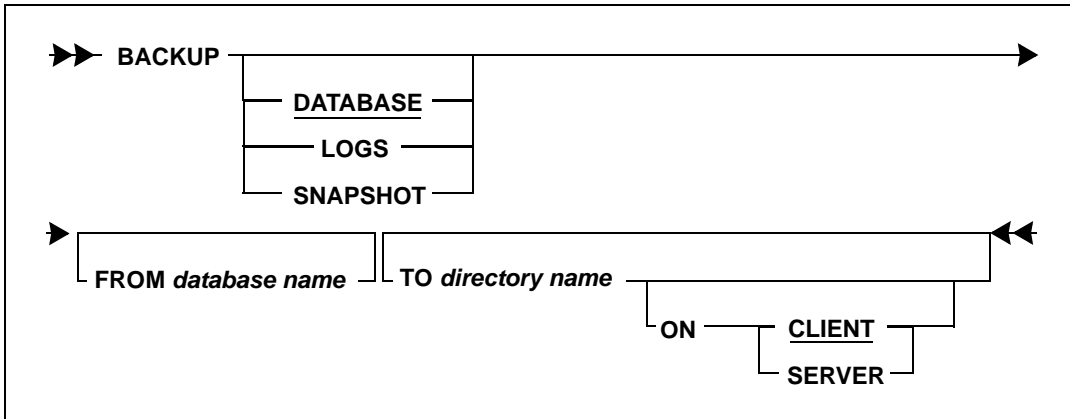
There are two ways of backing up a database: online and offline.

- An online backup is a copy of a database (*.dbs*) file that you make with a SQLTalk BACKUP command or a SQL/API function while the server program is running (users are connected to the database and transactions are in progress).
- An offline backup is a copy of the database file that you make with an operating system utility or command (such as *copy*) after successfully bringing the server down or de-installing the database.

### Online backups

The advantage of an online backup is that users can access the database while the backup is being done. This is important to sites which require the database to be up 24 hours a day.

The BACKUP command backs up a database, its transaction log files, or both.



*Syntax diagram of the BACKUP command*

**Note:** If you have a database greater than 2 gigabytes, you must perform the BACKUP command to multiple segments. This ability is provided in the BACKUP DATABASE option using the FROM and TO clauses. Read *Creating a segmented backup* on page 8-5.

The online backup options include:

- BACKUP SNAPSHOT (*sqlbss*)

This is a backup of the database and transaction log files at the moment that the backup begins. The BACKUP SNAPSHOT command backs up only the database file and those log files needed to restore the database to its current state. This includes the current active log file since BACKUP SNAPSHOT forces a log rollover.

This command is the only BACKUP command which does not require LOGBACKUP to be on.

- BACKUP DATABASE (*sqlbdb*)

This backs up the database file, without backing up the log files. It is used as the starting point for incremental backups. This backup is unusable if the accompanying log files are missing. You should *never* back up a database without also backing up the log files with it.

If your database exceeds 2 gigabytes in size, read the following section *Creating a segmented backup*.

- BACKUP LOGS (*sqlblf*)

This backs up the log files and then deletes them if they are not needed for transaction rollback or crash recovery.

BACKUP SNAPSHOT is used to freeze a database at a specific point in time. It is easy and provides you with a backup from which you can recover the database in one step. It does not, however, backup or recover any transactions that are made to the database after the snapshot is taken.

BACKUP DATABASE and BACKUP LOGS are provided for sites with large databases that wish to do incremental backups. Between database backups (both BACKUP SNAPSHOT and BACKUP DATABASE), you should back up log files using the BACKUP LOGS command or the *sqlb!f* API function. For example, you could back up the database and logs every Sunday, while on Monday through Saturday you could back up the logs.

A backup directory can be on a client or server computer. Once you have backed up a database and its log files to a directory, you can copy the backup files to archival media and delete the backup files from the client or server disk.

The examples later in this section show how to make and restore online backups.

## Creating a segmented backup

To perform segmented backups, you must create a control file (*databasename.BCF*) that describes the location and size of the segments to which you want to backup your database. To create a control file, use any editor on your system. The file must be in ASCII format.

The BACKUP command requires the name of the directory where the control file resides in the TO and FROM clause. If SQLBase detects a control file is present, a segmented backup operation is automatically performed. If the control file is not present, SQLBase backs up the database to a single *databasename.BKP* file which cannot exceed 2 gigabytes.

The backup control file follows this syntax:

```
FILEPREFIX <filename prefix>
DIR <destination dir> SIZE <file segment size>
DIR <destination dir> SIZE <file segment size>
```

This file provides the following information:

| <i>Parameter</i>         | <i>Description</i>                                             |
|--------------------------|----------------------------------------------------------------|
| <b><i>FILEPREFIX</i></b> | The prefix of the file segment names used for the load.        |
| <b><i>DIR</i></b>        | The destination directory where the load file segments reside. |

| Parameter | Description                                                                                                                                                                                                                                                        |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| SIZE      | The maximum size of the file segment in megabytes. You can specify a null or integer value of 1 through 2048 megabytes. The sum total of all size declarations in the control file must be large enough to contain the size of the database. (1MB=1,048,576 bytes) |

The following example shows a load control file located on a Windows NT/2000/XP/2003 server:

Example:

```
FILEPREFIX dbs
DIR c:\unldir\ SIZE 5
DIR d:\unldir\ SIZE 4
DIR e:\unldir\ SIZE 10
```

In this example, there are three file segments with the following characteristics:

- A segment called **c:\unldir\dbs.1** with 5 megabytes.
- A second segment called **d:\unldir\dbs.2** with 4 megabytes.
- A third segment called **e:\unldir\dbs.3** with 10 megabytes.

In this example, the sum total of the declared sizes is enough to contain a database no larger than 19 megabytes.

The name of this control file is *dbs.bcf*. SQLBase backs up the information from these three segments according to their listed order.

**Note:** If you are using a NetWare Server, be sure to specify both the volume mapping letter and the volume name for the file segments.

## Offline backups

An offline backup is a copy of the database file and log files that you make with an operating system utility or command (such as copy) after successfully bringing the server down or deinstalling the database.

The advantage of an offline backup is that you can back up files directly to archival media. For example, a SQLBase BACKUP command will not back up files to a tape drive.

Before you can make an offline backup, you must either deinstall the database or shut down the server, which will bring all of the databases on that server offline.

To shut down your server, please refer to *Chapter 6, Starting and stopping SQLBase*. You can also shut down a server by using the SQLBase Management Console, or by running the SHUTDOWN command in SQLTalk.

Use the SQLTalk DEINSTALL command or the *sqlded* API function to deinstall the database.

You make an offline backup using an operating system command or utility. Below is an example of an offline backup done using the COPY command:

```
C> COPY C:\Gupta\MYDB.DBS C:\BACKUPS\MYDB.BAK
C> COPY C:\Gupta\1.LOG C:\BACKUPS
C> COPY C:\Gupta\2.LOG C:\BACKUPS
```

Follow an offline backup with SQLTalk's SET NEXTLOG command to tell SQLBase that an offline backup of one or more log files has occurred. SQLBase now knows that these backed up log files are candidates for deletion. If you had backed up the log files with SQLBase's online BACKUP command, the files would have been automatically deleted. In the above case, the NEXTLOG command would be:

```
SET NEXTLOG 3;
```

## SQLTalk commands

SQLTalk commands for recovery are:

- SET RECOVERY ON/OFF

Recovery is on by default. This means that SQLBase logs all before and after images of changes to the database and creates log records for transaction control (checkpoints, for example). As part of transaction control, some log records document when a transaction started and how it ended (COMMIT or ROLLBACK). Transaction logs enable transaction rollback, crash recovery, and media recovery.

Recovery should be turned off only if you can afford to lose your data in the event of user error, a system crash, or a media failure. Once you set recovery off, some options are reset to their defaults including: autocommit, bulk execute, isolation level, load version, cursor-context preservation, restriction, rollback, scroll, and timeout.

You should back up your database and transaction logs before turning recovery off.

- SET LOGBACKUP ON/OFF (*sqlset* with the SQLPLBM parameter)

By default, LOGBACKUP is not enabled and SQLBase deletes log files as soon as they are not needed to perform transaction rollback or crash recovery. This is done so that log files do not accumulate and fill up the disk. If

LOGBACKUP is off, it is unlikely that you will be able to recover data created after your last successful backup of the database if it is damaged by a user error or a media failure.

If media recovery is important to your site, the LOGBACKUP parameter should be set on, specifying that logs are to be deleted by SQLBase only after they have been backed up.

This option is database specific and needs to be set on only once to avoid gaps in your log files. The setting will stay active until changed. You do not need to set the option each time a database is brought back online. Resetting this option affects whether log files are deleted or saved for archiving.

LOGBACKUP must be on to do a BACKUP DATABASE or BACKUP LOGS. It does not need to be on to do a BACKUP SNAPSHOT.

The default is off.

- SET CHECKPOINT (*sqlset* with the SQLPCTI parameter)

The checkpoint time interval parameter controls how often a recovery checkpoint operation is done. The default is every minute. Depending on the applications running against the server, a checkpoint operation can affect performance. Increasing the checkpoint interval may provide better performance. Longer checkpoint intervals reduce the impact on performance, but increase the time taken to perform crash recovery.

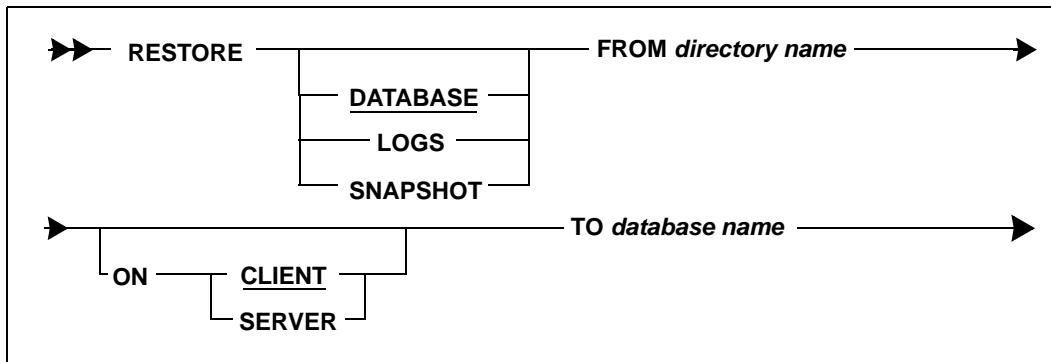
## Restoring a database

Data restoration is the process of recovering data which has been lost. Recovery of data is dependent on the DBA's backup planning and the frequency of backups made.

## Restoring an online backup

Users cannot be connected to a database during a restore and recovery process. You should deinstall a multi-user database with SQLTalk's DEINSTALL DATABASE command or the SQL/API's *sqlded* function, perform the restore and rollforward, and then re-install the database with SQLTalk's INSTALL DATABASE command or the SQL/API *sqlind* function.

If a database becomes damaged, you can restore a database backup with SQLTalk's RESTORE command or the SQL/API's *sqlrss* function.



*Syntax diagram of the RESTORE command*

**Note:** If you have a database greater than 2 gigabytes, you must perform the RESTORE command from multiple segments. This ability is provided in the RESTORE DATABASE option using the FROM clause. Read the following section, *Restoring from a segmented backup*.

The restore options include:

- RESTORE SNAPSHOT (*sqlrss*)

Executing a RESTORE SNAPSHOT command or calling the *sqlrss* function, copies the backup database file and the database log files to the database subdirectory and rolls back all uncommitted or incomplete transactions.

- RESTORE DATABASE (*sqlrdb*)

Executing a RESTORE DATABASE command or calling the *sqlrdb* function restores the backed up database only. It is unusable until the database log files are restored.

If you did not make the backup with BACKUP SNAPSHOT or you did a BACKUP SNAPSHOT and want to roll forward from that point to recover as much work as possible, you must give a RESTORE DATABASE command or call the SQL/API *sqlrdb* function in a program.

You must use this option if you are restoring a database from backup segments. This is required if you have a non-partitioned or partitioned database greater than 2 gigabytes. Each segment must be 2 gigabytes or less in size.

- RESTORE LOGS (*sqlrlf*)

Executing a RESTORE LOGS command or calling the *sqlrlf* function restores the backed up log files.

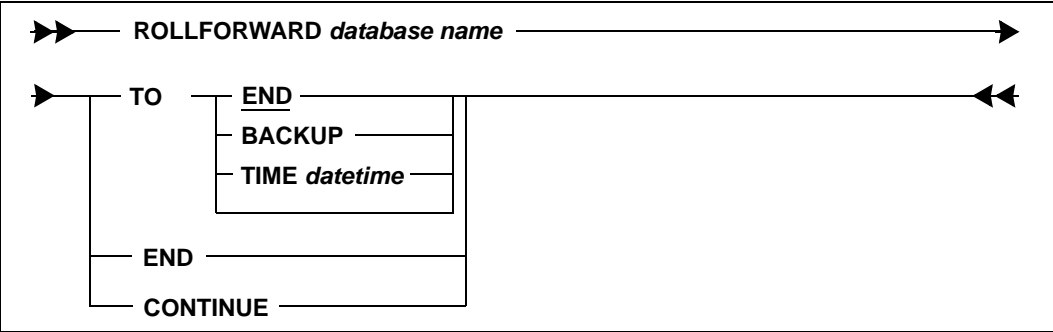
## Restoring from a segmented backup

To perform a restore from a segmented backup, you must specify a control file (*databasename.BCF*) that was used to create a successful segmented database backup. This control file describes the location and size of the segments from which you want to restore your database. For details on the backup control file, read *Creating a segmented backup* on page 8-5.

If SQLBase detects a control file is present, a segmented restore operation is automatically performed from the backup segments specified in the control file. If the control file is not present, SQLBase expects to restore the database from a single backup file that you specify named *databasename.BKP*, which must be less than 2 gigabytes in size

## Using ROLLFORWARD

To rollforward changes made after the database backup and bring the database up-to-date, use SQLTalk’s ROLLFORWARD command or the SQL/API’s *sqlrof* function.



*Syntax diagram of the ROLLFORWARD command*

The rollforward options are:

- **ROLLFORWARD TO END**  
Rolls forward through all log files available (the default). This recovers as much of the user's work as possible.
- **ROLLFORWARD TO BACKUP**  
Rolls forward to the end of the backup restored. This recovers all committed work up to the point when the database backup was completed. This is equivalent to a **RESTORE SNAPSHOT**.
- **ROLLFORWARD TO TIME**  
Rolls forward to a specified date and time. This allows you to recover a database up to a specific point in time, and in effect rolls back large “chunks”

of committed and logged work that you no longer want applied to the database. For example, if data is erroneously entered into the database, you would want to restore the database to the state it was in before the bad data was entered.

You must have backed up *all* the database's log files and must apply them *in order* or the rollforward will fail. If you are missing any of the log files, you will not be able to continue rolling forward from the point of the last consecutive log. For example, if you have *1.log*, *2.log*, *4.log*, and *5.log*, but *3.log* is missing, you will only be able to recover the work logged up to *2.log*. *4.log* and *5.log* cannot be applied to the database. An unbroken sequence of log files is required to recover a database backup to its most recent state.

The rollforward operation stops if SQLBase cannot find a log file that it needs. In this situation, a RESTORE LOGS command will be needed to copy the log files needed from the backup directory to the database directory.

If there are more logs to be processed than can fit on disk at one time, you can give the RESTORE LOGS command repeatedly to process all the necessary logs.

If a log file requested is not available, you can enter a ROLLFORWARD *database* END command to end recovery using the data restored up to that point. The *sqlenr* API function performs the same operation.

## Restoring an offline backup

Recovering an offline backup is done in one of two ways:

- If the backup consists of only a database file, restore it by copying it over the existing database file, making sure the extension is *.dbs* (you may have changed it, for example, to *.bkp* when you backed it up), and then connecting to the database. All changes made since the offline backup was done will be lost.
- If the backup consists of a database file and one or more log files, use the SQLTalk RESTORE DATABASE command or *sqlrdb* API function to restore the database and then give a ROLLFORWARD command to apply the logs to bring it up-to-date. The RESTORE copies the backup to the database subdirectory, and the ROLLFORWARD applies the committed and logged changes made to the database since the offline backup of the database was taken.

If SQLBase cannot find the log files to rollforward, you can restore them by either a RESTORE LOGS command (which automatically does a ROLLFORWARD CONTINUE) or with a copy command or utility, followed by a ROLLFORWARD CONTINUE command, or call the *sqlcrf* API function to apply the log files.

In order for the RESTORE command to work, the name of the database backup file must be *database.BKP*.

## Examples

This section provides backup and restore examples.

### Example 1 - Snapshot

The following example shows you the recommended way to back up (BACKUP SNAPSHOT) and restore (RESTORE SNAPSHOT) a database and its logs. Snapshots are easy because you can backup a database or restore it with one command. A restore snapshot is a complete transaction, and should not be followed by further data restoration such as RESTORE LOGS.

Recall that BACKUP SNAPSHOT is the only backup command which does not require LOGBACKUP to be on.

1. Establish a connection to the server:

```
SET SERVER SERVERP/PPPPP;
SERVER IS SET
```

2. Create a subdirectory on the client where you will store the backup, then back up the *.dbs* and *.log* files to the client computer:

```
BACKUP SNAPSHOT TO \BACKUP\DEMOX ON CLIENT;
SNAPSHOT BACKED UP
```

You are now ready to restore the database and its log files.

3. Restore the database and its log files from the client computer:

```
RESTORE SNAPSHOT FROM \BACKUP\DEMOX ON CLIENT TO DEMOX;
SNAPSHOT RESTORED
```

### Example 2 - Backing up and restoring database and log files separately

This example shows how to backup and restore a database and its log files separately. You should *always* backup the transaction log files as well as the database. Remember that LOGBACKUP must be on.

1. Establish a connection to the server:

```
SET SERVER SERVERP/PPPPP;
SERVER IS SET
```

2. Create a subdirectory on the client where you will store the backup:

3. Back up the database and log files to the client computer:

```
BACKUP DATABASE TO \BACKUP\DEMOX ON CLIENT;
DATABASE BACKED UP
```

4. Force a log rollover:

```
RELEASE LOG;
RELEASE LOG COMPLETED

BACKUP LOGS TO \BACKUP\DEMOX ON CLIENT;
3 LOGS BACKED UP
```

Now you are ready to restore the database and its log files

5. Restore the database from the client computer and apply the transactions in the log files to the database using the ROLLFORWARD command:

```
RESTORE DATABASE FROM \BACKUP\DEMOX TO DEMOX;
DATABASE RESTORED
```

6. Use the ROLLFORWARD command to initiate recovery for this restored database.

```
ROLLFORWARD DEMOX;
ROLLFORWARD STARTED
```

The log file 1.LOG could not be found. Use the RESTORE LOGS command to restore this log and continue the rollforward process. If this log is not available, use the ROLLFORWARD <database> END command to complete the recovery process.

```
RESTORE LOGS FROM \BACKUP\DEMOX TO DEMOX;
3 LOGS RESTORED
```

The log file 4.LOG could not be found. Use the RESTORE LOGS command to restore this log and continue the rollforward process. If this log is not available, use the ROLLFORWARD <database> END command to complete the recovery process.

```
ROLLFORWARD DEMOX END;
ROLLFORWARD COMPLETED
```

## Example 3 - Performing Incremental Backups with Backup Snapshot

This example shows how to perform and restore incremental backup using BACKUP SNAPSHOT followed by BACKUP LOGS. Remember that BACKUP LOGS requires LOGBACKUP to be ON.

In this example, the BACKUP SNAPSHOT is performed at 12:00 am. Monday.

1. Establish a connection to the server.

```
SET SERVER SERVERP/PPPP;
SERVER IS SET.
```

2. Create a sub-directory where the backup will be stored.

```
\MON_BASE
```

3. Backup the database and log files to the client computer.

```
BACKUP SNAPSHOT FROM MYDB TO \MON_BASE ON CLIENT;
SNAPSHOT BACKED UP.
```

At 10:00 am Monday, the first incremental backup is performed:

1. Establish a connection to the server.

```
SET SERVER SERVERP/PPPP;
SERVER IS SET.
```

2. Create a sub-directory where the backup will be stored.

```
\MON_1000
```

3. Backup the log files to the client computer.

```
BACKUP LOGS FROM MYDB TO \MON_1000;
n LOGS BACKED UP
```

At 2:00 pm Monday, the second incremental backup is performed.

1. Establish a connection to the server.

```
SET SERVER SERVERP/PPPP;
SERVER IS SET.
```

2. Create a sub-directory where the backup will be stored.

```
\MON_1400
```

3. Backup the log files to the client computer.

```
BACKUP LOGS FROM MYDB TO \MON_1400;
n LOGS BACKED UP
```

At 5:00 pm, an operating system failure causes the loss of the database. By restoring with incremental backups, as much data as possible can be recovered.

1. Create a subdirectory.

```
\MON_1700
```

2. Copy all logs from %DBDIR%\MYDB or %LOGDIR%\MYDB to the directory \MON\_1700.

This backs up all existing logs made since 2:00pm.

3. Create a subdirectory to restore from.

```
\MYRESTORE
```

4. Copy the logs to that directory.

```
COPY FROM \MON_BASE TO \MYRESTORE
```

```
COPY FROM \MON_1000 TO \MYRESTORE
```

This overwrites any files of the same name.

```
COPY FROM \MON_1400 TO \MYRESTORE
```

This overwrites any files of the same name.

```
COPY FROM \MON_1700 TO \MYRESTORE
```

This overwrites any files of the same name.

5. Restore the database. This requires a server connection.

```
RESTORE DATABASE FROM \MYRESTORE TO MYDB
```

6. Restore the logs

```
RESTORE LOGS FROM \MYRESTORE TO MYDB
```

7. Initiate ROLLFORWARD through the restored logs.

```
ROLLFORWARD DATABASE MYDB;
```

8. Complete the database rollforward.

```
ROLLFORWARD MYDB END;
```

The database is now at a state of consistency at 5:00 pm Monday. All committed transactions are rolled in and all uncommitted transactions are rolled back.

## Backing up and restoring partitioned databases

You can back up large partitioned databases using SQLTalk's BACKUP, RESTORE, and ROLLFORWARD commands. If your database is greater than 2 gigabytes, you need to create a segmented backup database. Refer to the sections *Creating a segmented backup* and *Restoring to a segmented database* in this chapter for details.

### Online backup

Following is an example for converting a non-partitioned database to a partitioned database using the online BACKUP and RESTORE commands.

1. Back up the current DEMOX database. This step assumes you have already entered a SET SERVER command.

```
BACKUP DATABASE TO \BACKUP\DEMOX;
DATABASE BACKED UP
```

2. Drop the old DEMOX database.

```
DROP DEMOX;
```

3. Create the new backup database.

```
CREATE DATABASE DEMOX;
```

Note that in this step, you may need to create a partitioned database. For details, read *Partitioning a database* on page 6-17

4. Restore the DEMOX database.

```
RESTORE DATABASE FROM \BACKUP\DEMOX TO DEMOX;
DATABASE RESTORED
```

---

**Note:** When backing up and restoring a partitioned database, you also need to backup and restore the MAIN database. For details, see the *The MAIN database* on page 8-17.

---

### Offline backup

You can also create an offline backup of a partitioned database. To do this, first be sure to shut down the server gracefully. Then use your operating system command to back up the dbareas that make up your partitioned database. Also be sure to make backups of the MAIN database and its logs. The MAIN database is described in the next section.

If you want to restore an offline backup, you can restore all relevant dbareas and the MAIN database, including its files, in the appropriate locations.

## The MAIN database

The MAIN database contains information about the partitioned databases. Because it is small, it is easiest to back it up online.

---

**Note:** *Always back up the MAIN database when you back up a partitioned database.*

---

To restore the MAIN database, first disable partitioned databases with the following command:

```
SET PARTITIONS OFF;
```

Once recovery is complete, re-enable partitioned databases with the following command:

```
SET PARTITIONS ON;
```

You must restore the MAIN database by rolling forward through all the log files that were backed up when the MAIN database itself was backed up. You cannot do a partial recovery of the MAIN database.

Do not delete the MAIN database.

## Backups with secure databases

For information on backing up secure databases, read *Chapter 7, Security and Authorization*.



## Chapter 9

# Distributed Transactions

---

This chapter describes how to create distributed transactions. Distributed transactions in SQLBase use the standard two-phase commit protocol. This protocol maintains transaction integrity and communication among the individual databases involved in a distributed transaction.

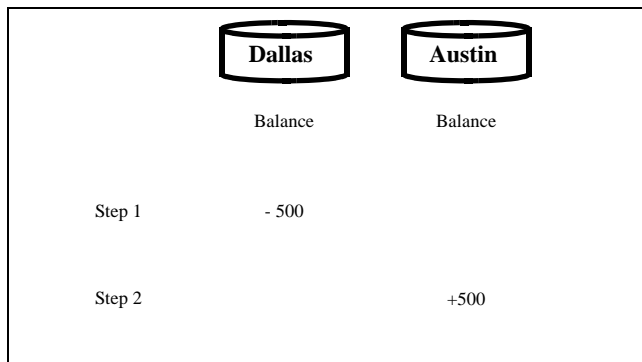
## What is a distributed transaction?

A distributed transaction coordinates SQL statements among multiple databases that are connected by a network. The databases that participate in a distributed transaction can reside anywhere on the network.

In a distributed transaction, the coordinating application communicates among the participant databases and verifies data integrity. It maintains this integrity even when a crash occurs.

A distributed transaction conforms to the same data consistency rules as a single database transaction — either all of the transaction's statements commit, or none at all.

A distributed transaction can be necessary for efficient transaction management. For example, assume a bank in Dallas needs to transfer \$500 to its Austin branch.



The application uses the following steps:

1. Debit \$500 from the Dallas branch.
2. Credit the \$500 to the Austin branch.

It is possible to use regular transactions here by just sending a COMMIT message to both branches after step 2 and waiting for their replies. However, severe data integrity problems could result if a system/network error prevents either site from completing their part of the transaction:

- If a network or system failure occurs at the Dallas site after step 1 but before step 2, the \$500 would not be debited. Unaware of this problem, the Austin branch would commit its \$500 deposit. The customer would end up with two deposits.
- If a network or system failure occurs at the Austin site before step 2, the deposit would not be credited to the Austin branch. Unaware of the problem,

Dallas would commit the \$500 debit. The bank would have no record of the customer's deposit at all.

A distributed transaction using two-phase commit eliminates these data integrity problems by coordinating communication between sites throughout the various transaction stages.

## Setting up a distributed transaction

You can set up a distributed transaction either in SQLTalk or with a SQL/API function. You can also create a distributed transaction through Gupta's Team Developer product, although that is not documented in this chapter.

Server connects (*sqlcsv*) and connects with recovery turned off cannot participate in a distributed transaction. In addition, an application cannot connect to a database in both distributed and non-distributed transaction mode.

In a distributed transaction, one of the participating database servers must also be the *commit server*. The commit server logs information about the distributed transaction and assists in recovery after a network failure. To enable commit server capability for a server, set the *commitserver* keyword to 1 (on) in SQL.INI. Read *Components* on page 9-7 for more information on the commit server.

Databases participating in a distributed transaction must conform to the following communication requirements:

- They must reside on the same network. They can reside on the same server, but this is not required.
- Each participating database server that has commit service enabled must be able to connect to all other servers involved in the distributed transaction. If all the servers have commit service capability, they all must be able to connect with each other.
- If you are using Novell's NetWare, specify the [nwclient] section for each server that is participating in a distributed transaction. This allows servers to communicate mutually. Communication between servers only occurs when a commit server:
  - Verifies it can talk to all other participating servers at the time of a distributed commit. (This is performed at most once per participant.)
  - Attempts to contact other participating servers under a failure condition.

Using SQLTalk

In SQLTalk, set the DISTRANS option on to start a distributed transaction. Any new database you connect to is then part of a distributed transaction.

The following example sets up a distributed transaction between the Austin and Dallas sites in a sample SQLTalk session:

Notice that you need to disconnect the first cursor in order to use it in the distributed transaction, since it was initially connected in a regular transaction.

```
set distrans on;
DISTRIBUTED TRANSACTION MODE IS NOW ON
connect austin 2;
CURSOR 2 CONNECTED TO AUSTIN
disconnect 1;
CURSOR 1 DISCONNECTED
connect dallas 1;
CURSOR 1 CONNECTED TO DALLAS
select * from account where account_num=14560;

NAME ACCOUNT_NUM BALANCE
=====
Nanda Smith 14560 1000

1 ROW SELECTED

update account set balance=balance-500 where account_num=14560;
1 ROW UPDATED
use 2;
select * from account where account_num=14560;

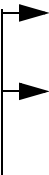
NAME ACCOUNT_NUM BALANCE
=====
Nanda Smith 14560 1000

1 ROW SELECTED

update account set balance=balance+500 where account_num=14560;
1 ROW UPDATED
commit;
TRANSACTION COMMITTED
use 1;
select * from account where account_num=14560;

NAME ACCOUNT_NUM BA
=====
Nanda Smith 14560 500

1 ROW SELECTED
```



Note here that only one COMMIT statement is issued. In a regular transaction, you would need to issue two COMMIT statements, one on each database.

```

use 2;
select * from account where account_num=14560;
NAME ACCOUNT_NUM BALANCE
=====
Nanda Smith 14560 600

1 ROW SELECTED
exit;

```

## Using the SQL/API

In the SQL/API, use the *sqlset* function in conjunction with the SQLPDTR parameter to set distributed transaction mode on. Once you set this parameter on, all subsequent commands automatically become part of a distributed transaction. Read the *Application Programming Interface* manual for an example.

## Other issues

This section describes how creating distributed transactions affects SQLBase in general.

### Backing up and restoring

You cannot back up and restore database snapshots spanning more than one database.

### Isolation

Changing the isolation level of a distributed transaction changes it for all participating databases. Also, since changing isolation levels results in an implicit commit, SQLBase issues a coordinated commit when you are using a distributed transaction.

### Lock wait timeout

To set a specified lock wait timeout for a distributed transaction, use the *sqlset* function with the SQLPWTO parameter. Setting the timeout for a distributed transaction sets it for all participating cursors. This is the mechanism to get out of a global lock in a distributed transaction.

### Savepoints

When you set a savepoint in a distributed transaction, it applies to all the databases which participate in that transaction. For example, assume that the Austin database contains a savepoint. A rollback to the savepoint rolls back actions on both the Austin and Dallas databases.

```

set distrans on;
DISTRIBUTED TRANSACTION MODE IS NOW ON
connect austin 2;
CURSOR 2 CONNECTED TO AUSTIN

```

```

disconnect 1;
CURSOR 1 DISCONNECTED
connect dallas 1;
CURSOR 1 CONNECTED TO DALLAS

```

- (1) ➔ **insert into account values.....**  
 use 2;+  
 set savepoint savpoint1;  
 use 1;  
 insert into account values ....
- (2) ➔ **select \* from account where account\_num=14560;**  
 1 ROW SELECTED  
 use 2;  
 insert into account values ...
- (3) ➔ **rollback savpoint1;**  
**commit;**

In this scenario, statements 2 and 3 are rolled back, but statement 1 is committed.

## AUTOCOMMIT

Turning on AUTOCOMMIT for a cursor in a regular transaction causes an implicit commit after each command. For a cursor in a distributed transaction, AUTOCOMMIT causes an implicit coordinated commit every time that cursor gets executed.

The SQLBase OLE DB Data Provider is itself transaction-based. While using this provider, if you turn off AUTOCOMMIT, or turn it on, each of these changes starts a new transaction. If there was uncommitted work remaining when this setting change is attempted, the change will fail with an error. Be sure to explicitly COMMIT or ROLLBACK such work before attempting to change the AUTOCOMMIT setting.

## Last disconnect

For regular transactions, the last disconnect from a database results in an implicit commit. For distributed transactions, however, the last disconnect from any participating database returns an error if there is any uncommitted work, such as modifications that have been neither committed nor rolled back.

## Recovery

You cannot set recovery off in a distributed transaction.

# Two-phase commit

The two-phase commit protocol coordinates a transaction commit process on all participating databases. When a network or system failure occurs, the transaction is resolved rather than left in a partially committed state. Without two-phase commit,

you could end up with a partially committed distributed transaction, which is committed on one system, and rolled back on another.

Understanding the basic mechanics of a two-phase commit can help you plan efficient database access, especially if you need to manually resolve a transaction if a server crashes.

## Components

Two-phase commit uses the following components:

- coordinator
- participant
- commit server

### Coordinator

The application that initiates a distributed transaction is called the *coordinator*. The coordinator is responsible for starting the two-phase commit, communicating among the participants, and logging transaction information with the commit server.

The actual code for the coordinator resides in Gupta's SQL/API, not in the coordinating application. You do not need to write any code yourself to enable coordinator responsibilities.

In the bank example, the transfer program is the coordinator.

### Participant

The databases involved in the transaction are called *participants*, and can exist at different sites. In the bank example, the participants are the Dallas and Austin branches.

The participants should either reside on the same network, or be able to communicate across the network. This is important in a failure recovery situation, which is discussed later in the chapter.

Each participant has its own SYSADM.SYSPARTTRANS table, where it records information about the status of the transaction. This table contains information about transactions which are *in-doubt* (in an undecided state) after crash recovery or data restoration. There is one row for each in-doubt transaction. Once a transaction is resolved, the information is removed from the table. See the following section on *Failure recovery* for more information on in-doubt transactions.

### Commit server

Two-phase commit uses a *commit server* that logs information about each distributed transaction. It performs the following functions:

- Assigns a global ID to the transaction. This global ID distinguishes a distributed transaction from a regular transaction.
- Starts and ends a distributed transaction.
- Commits or rolls back a distributed transaction.
- Logs the status of a distributed transaction throughout its various stages.
- Assists in recovery in case of a crash.

The commit server logs information about the distributed transaction in two MAIN database tables:

- `SYSADM.CSVTRANS`

This table records general information about a transaction such as the global ID, number of participants, and current state of the transaction.

- `SYSADM.CSVPARTS`

This table contains information about all the participants in a transaction, such as database name and user name.

The information in these two tables help in the recovery process if your system crashes. This is described in the *Failure recovery* section later in this chapter.

---

**Note:** If a database server is designated to be the commit server for a distributed transaction, a MAIN database is created for that server even if it is non-partitioned.

---

The commit server only retains information about a transaction while it is being processed. After a transaction is resolved (COMMIT or ROLLBACK), the commit server deletes all information about the transaction from the two systems tables.

You cannot create a distributed transaction unless one of the participating database servers has commit service enabled. To enable commit server capability for a server, set the *commitserver* keyword to 1 (on) in the database server's SQL.INI file.

Generally, there is only one participant database server with commit service enabled. However, if the distributed transaction connects to multiple database servers that have commit service capability, the coordinator randomly chooses one to be the commit server for the transaction.

## Process

There are two stages in the two-phase commit process: prepare and commit. In the first stage, the coordinator sends a PREPARE message to all the participants. Each participants decides individually whether it can commit the transaction, or needs to rollback.

If the participant cannot commit the transaction, it rolls back its part of the transaction, and sends a ROLLBACK vote to the coordinator. If the coordinator receives even one ROLLBACK vote, it instructs the other participants to rollback the transaction.

If the participant can commit the transaction, it sends a COMMIT vote to the coordinator. Once a participant votes to commit, it cannot later decide to rollback the transaction.

If the coordinator receives COMMIT votes from all the participants, it sends them all an instruction to go ahead and commit the transaction. The participants commit the transaction accordingly.

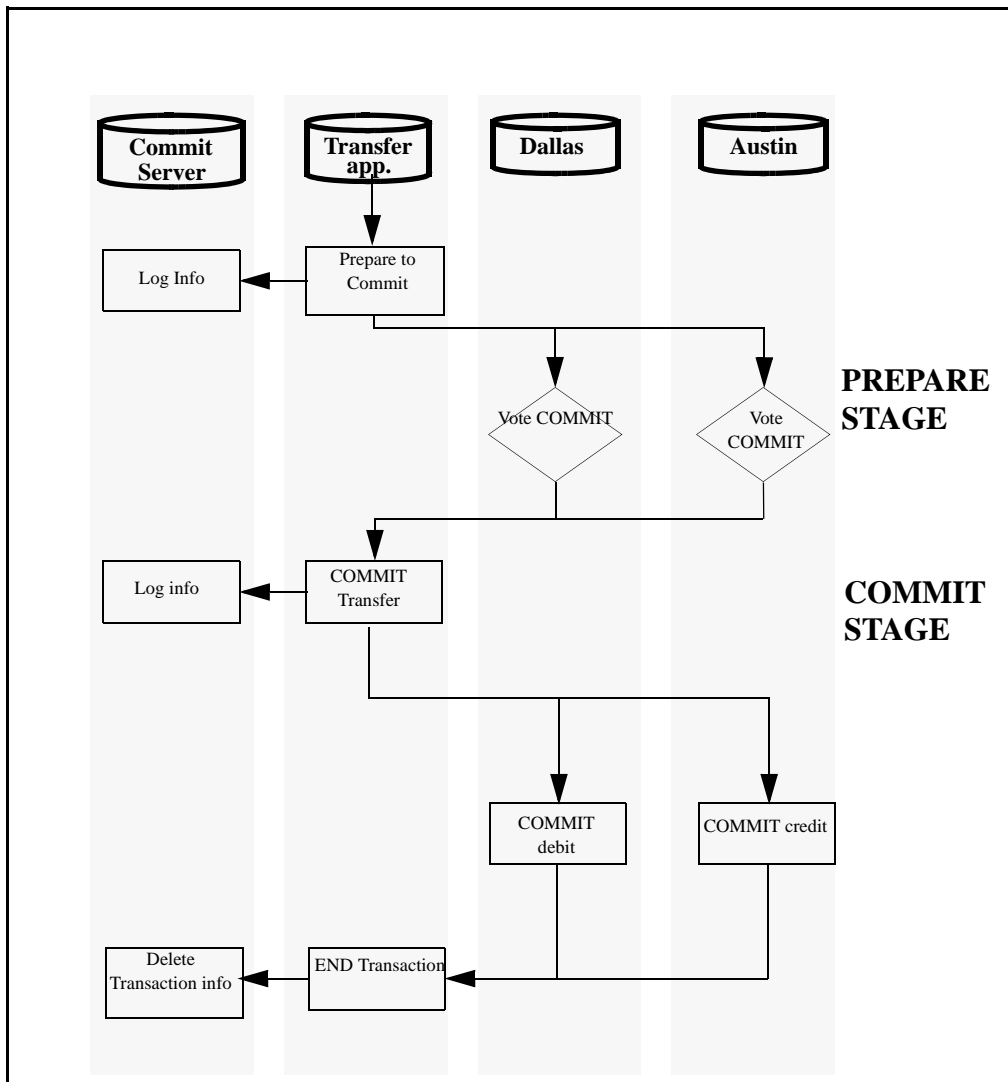
The Commit Server logs information about the transaction during the two-phase commit process.

After a commit or rollback, the system writes a commit log record and releases all the transaction's locks. The transaction entry is also removed from SYSADM.CSVTRANS, SYSADM.CSVPARTS, and the SYSADM.SYSPARTTRANS tables in the participating databases.

The following figures use the bank example to demonstrate the two-phase commit process. The first figure shows a successfully committed transaction; the second demonstrates what happens when one of the participants aborts.

## COMMIT Example

In this figure, the participants (Dallas and Austin branches) successfully commit the transfer process. The transfer application is the coordinator.



1. In the PREPARE stage, the coordinator (transfer program) sends a message to the Dallas and Austin databases to prepare a commit.

Both Austin and Dallas vote to commit the transaction. They are now in an in-doubt state until they actually receive the COMMIT message from the coordinator.

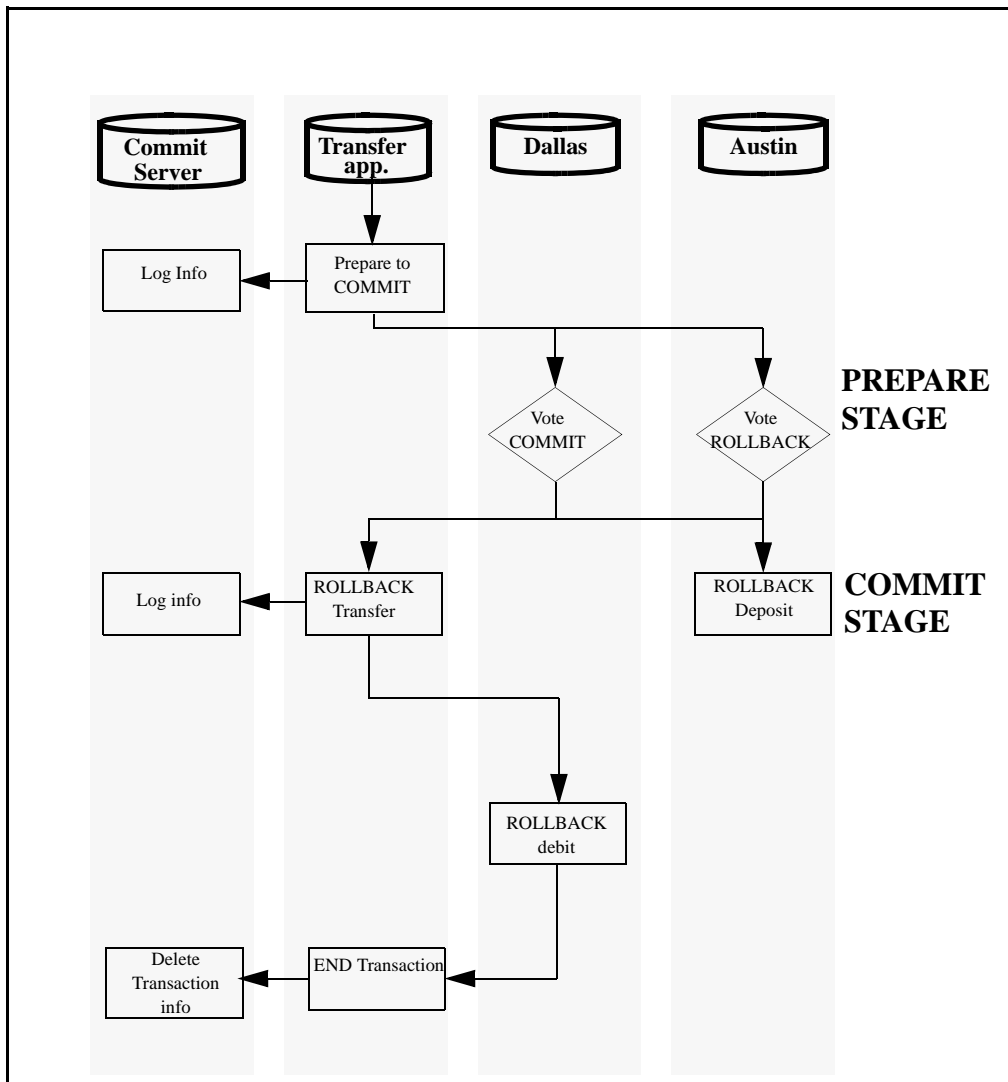
2. Since it received commit votes from all the participants, the coordinator begins the commit process. It sends a COMMIT message to both Austin and Dallas. Austin and Dallas both commit their part of the transaction.

When they complete the COMMIT, both branches inform the transfer program, and delete information about the transaction in their local SYSADM.SYSPARTTRANS table. The program logs this information with the commit server. The commit server then deletes the information about the transaction in both SYSADM.CSVTRANS and SYSADM.CSVPARTS in the MAIN database.

For a distributed transaction, a commit operation results in a coordinated commit across all the databases. This is also true for an implicit commit.

### ROLLBACK example

In this example, the Austin branch is unable to commit the transfer. The following figure demonstrates this process.



In this example, Austin cannot commit the transaction, and votes to rollback. The coordinator accordingly sends a ROLLBACK message to Dallas.

A rollback operation rolls back the changes to all the databases, including implicit rollbacks. For example, if a transaction needs to be rolled back due to an error, the rollback is performed on all the databases to which that transaction is connected.

# Failure recovery

A network or system failure can cause special problems for databases using distributed transactions, such as partially committed transactions, locking of needed data, and data inconsistency.

## In-doubt transactions

If the database or server crashes, the database is recovered when you reconnect. At this point, there may be in-doubt transactions that need to be resolved. A transaction is in-doubt when it is ready to commit a transaction, but has not yet received a COMMIT or ROLLBACK message from the coordinator; it is unknown which of the transactions committed and which did not. To resolve these transactions automatically when you reboot, SQLBase uses the *commit server daemon*.

For example, assume that the bank transfer program crashes after Dallas commits its delete command, but before Austin records the transfer. The Dallas transaction is resolved, but the Austin transaction is in-doubt. It remains in-doubt until the commit server daemon can resolve it after you reboot the system.

## Manually resolving transactions

You should always allow SQLBase to automatically recover in-doubt transactions. However, if the network connections are damaged, or the server will take too long to recover, occasionally you may need to manually commit or rollback the transaction if it is locking data required by other users, or pinning important log files.

---

**Note:** If you do decide to manually override a transaction, first consult with the administrators at the other locations. Be aware that if you make a wrong decision, you may create data inconsistencies that must be manually corrected, and may be difficult to trace.

---

Use the following guidelines for resolving in-doubt transactions yourself:

1. Find out which transactions are in-doubt.
2. Find the status of these in-doubt transactions if possible.
3. Force the resolution.

## Finding in-doubt transactions

To locate in-doubt transactions query the local SYSADM.SYSPARTTRANS table. Note the global ID numbers of the transactions listed, and the name of the commit server.

It is important to remember that this table does not contain information about all current in-doubt transactions while the database server is active. You only find out about all the transactions when you reconnect after the crash.

## Finding transaction status

If only the network connection crashed and you can still access the commit server locally, locate the commit server and try to access it. Then, use the global ID from `SYSADM.SYSPARTTRANS` to find the transaction in the `SYSADM.CSVTRANS` table in the MAIN database. Find the row that corresponds to your transaction ID. The `STATE` column tells you the transaction status for that record. It can be one of the following:

- IDLE
- PREPARED
- COMMITTING
- ABORTING

If the transaction status is `COMMITTING`, commit the transaction on the local database. If it is any other state, abort it.

If the commit server itself is damaged and the `SYSADM.CSVTRANS` table is inaccessible, you need to decide yourself how to resolve the in-doubt transaction. Here are some suggestions:

- Call the administrators at the other sites and find out the state of their transaction. Using the bank situation as an example, if the Dallas branch committed a debit of \$500, the Austin branch should go ahead and commit the deposit record.
- Check the data affected by the transaction. For example, if the Dallas branch shows a debit of \$500, it may be waiting for a final `COMMIT` message from the coordinator. You can manually force this commit yourself.

## Forcing the resolution

After deciding how to resolve the transaction, use the following commands to force either a commit or rollback:

```
COMMIT TRANSACTION <transaction-id> FORCE
```

or

```
ROLLBACK TRANSACTION <transaction-id> FORCE
```

After you force the resolution, all information about the transaction is deleted from the `SYSADM.SYSPARTTRANS`, `CSVTRANS`, and `CSVPARTS` tables automatically.

Both these commands require DBA privilege.

## Performance issues

While two-phase commit is critical to guaranteeing distributed transaction integrity, realize that it does generate network traffic and can affect your database performance.

If there are  $X$  number of systems involved, the coordinator must send and receive at least  $4X$  messages to commit the transaction. This is in addition to the messages that carry the SQL statements and query results. A result of this network overhead can be a decrease in database performance.

In addition, the coordinator must open a second cursor to each database to send the two-phase commit messages.

To improve performance, your database design should maintain frequently accessed data on one system as much as possible. And while a distributed transaction can be the most efficient way to carry out a command, consider the performance issues carefully before using it.



## Chapter 10

# National Language Support

---

SQLBase supports English as its standard language, but it also supports many international languages including those spoken in Europe and Asia. This chapter briefly describes how to configure SQLBase to support languages other than English.

## NLS configuration files

There are six files that you need to work with in enabling SQLBase to support a language other than English:

- SQL.INI
- country.sql
- country.dbs
- error.sql
- country.tlk
- message.sql

### SQL.INI

In order to support a non-English language, you need to configure a *country* keyword in on both the client and the server.

The country keyword instructs SQLBase to use the settings in the specified section of the country.sql file. For example, if you want your database server to support germany, specify:

```
[dbntsrv]
...
country=germany
```

The matching entry in the country.sql file on the server looks like this:

```
[germany]
keyword1=value
keyword2=value
```

You also need to specify the country keyword on the client:

```
[win32client]
...
country=germany
...
; add next section if it does not exist
[sqltalk]
country=germany
```

Read *Chapter 3, Configuration: SQL.INI and the Registry* for detailed information about the country keyword.

## country.sql

The country.sql file is designed to hold most of the information necessary to customize SQLBase for non-English languages.

This file contains a section for each country that SQLBase supports and each section name is the name of a country.

For example, the country.sql file contains the following sections:

```
[france]

[japan]

[germany]
```

The section name corresponding to the country you want SQLBase to support should match the value of the configuration file's (SQL.INI) country keyword.

### Directives

Each section contains statements called directives which specify options and settings. Each directive starts with a pound sign (#) followed by the directive name as well as data that SQLBase can use to perform operations dependent on language differences. A description of each directive follows.

The data that you specify after a directive contains standard characters or character strings which you can edit with a standard text editor. Specify special or non-printable characters in hexadecimal notation: \hex-number. The back-slash (\) character means that a hexadecimal number follows. The number is the hexadecimal (binary) representation of a character.

### Code page

Some (but not all) of the directives combine together to make up the code page. The code page is a 256-byte table in memory that describes the attributes of each character.

SQLBase starts out with a default code page that is used for standard English. If you specify any one of the code page directives, SQLBase redefines the *entire* code page. Therefore, if you specify one of the code page directives then you must specify all of the code page directives. The default code page is completely overwritten with new information specified by code page directives.

The following table shows the directives that affect the code page and those that do not.

| Affect the code page                                                                   | Do not affect the code page           |
|----------------------------------------------------------------------------------------|---------------------------------------|
| alpha<br>firstbyte<br>identifier<br>numeric<br>punctuation<br>secondbyte<br>whitespace | lower<br>prefix<br>translate<br>upper |

Sorting

SQLBase uses a Sort-Merge algorithm to sort the default English settings. For non-English sorting, SQLBase uses an alternative sorting algorithm to deal with special ASCII/ANSI sort orders. This switch is done automatically when SQLBase detects a national language setting in country.sql.

country.dbs and country.tlk

Both files are required to build a database with National Language Support. country.tlk is a script that is applied to country.dbs to create start.dbs. *Building a database with NLS* on page 10-9 explains how to use country.dbs and country.tlk to build a database with National Language Support.

error.sql

All Gupta client and server software need this error file. Besides translating the text of each error message from English to a non-English language, you should also translate the reason and remedy explanations for each error message.

message.sql

All Gupta client and server software need this message file too. It contains prompts, confirmations, and other non-error messages for Gupta software.

# country.sql directives

This section describes, and provides an example of, each directive.

## #alpha

### Description

This directive specifies the characters that are alphabetic. Alphabetic characters are significant when SQLBase parses identifiers.

Follow this directive with a list of characters.

### Example

The country Japan could have this *alpha* directive:

```
[Japan]
#alpha
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
```

## #firstbyte

### Description

This directive specifies the characters that are allowed as the first byte of double-byte characters. Only use this directive for countries which use a double-byte character set.

Follow this directive with a list of characters.

### Example

The country Japan could have this *firstbyte* directive:

```
[Japan]
#firstbyte
\81 \82 \83 \84 \85 \86 \87 \88 \89 \8A \8B \8C \8D \8E \8F
\90 \91 \92 \93 \94 \95 \96 \97 \98 \99 \9A \9B \9C \9D \9E \9F
\E0 \E1 \E2 \E3 \E4 \E5 \E6 \E7 \E8 \E9 \EA \EB \EC \ED \EE
\EF \F0 \F1 \F2 \F3 \F4 \F5 \F6 \F7 \F8 \F9 \FA \FB
```

## #identifier

### Description

This directive specifies additional characters that are allowed in identifiers, such as column names, table names, and index names. These characters are in addition to the standard alpha-numeric characters allowed in identifiers.

Follow this directive with a list of characters.

### Example

The country Japan could have this identifier directive:

```
[Japan]
#identifier
 \AE \AF
\B0 \B1 \B2 \B3 \B4 \B5 \B6 \B7 \B8 \B9 \BA \BB \BC \BD \BE \BF
\C0 \C1 \C2 \C3 \C4 \C5 \C6 \C7 \C8 \C9 \CA \CB \CC \CD \CE \CF
\D0 \D1 \D2 \D3 \D4 \D5 \D6 \D7 \D8 \D9 \DA \DB \DC \DD \DE \DF
```

### #lower

#### Description

Some countries require special handling when converting a string to lowercase. SQLBase automatically supports standard English lowercase conversion. The lower directive alters the lowercase conversion to match the needs of a country.

#### Example

If the country Glarpia needs to convert “X” to “n” and “Y” to “m” when doing a lowercase operation, the lower directive would be:

```
[Glarpia]
#lower
X n
Y m
```

### #numeric

#### Description

This directive specifies the characters that are numeric. Numeric characters are significant when SQLBase parses identifiers.

Follow this directive with a list of characters.

#### Example

The country Japan could have this numeric directive:

```
[Japan]
#numeric
0 1 2 3 4 5 6 7 8 9
```

## #prefix

### Description

This directive specifies prefixes that SQLBase should ignore when collating data. (A prefix to a string is the front part of the string. For example, if 'La' and 'Le' are described as prefixes, then 'La SQL' and 'Le SQL' would collate in the same position.)

Follow this directive with a list of prefixes.

### Example

If SQLBase should ignore the prefixes "LA" and "LE" in the country Glarpia, the prefix directive would be:

```
[Glarpia]
#prefix
LA
LE
```

## #punctuation

### Description

This directive specifies punctuation characters.

Follow this directive with a list of characters.

### Example

The country Japan could have this punctuation directive:

```
[Japan]
#punctuation
\A1 \A2 \A3 \A4 \A5
```

## #secondbyte

### Description

This directive specifies characters that are allowed as the second byte of double-byte characters. Only use this directive for countries which use a double-byte character set.

Follow this directive with a list of characters.

## Example

The country Japan could have this secondbyte directive:

```
[Japan]
#secondbyte
\40 \41 \42 \43 \44 \45 \46 \47 \48 \49 \4A \4B \4C \4D \4E \4F
\50 \51 \52 \53 \54 \55 \56 \57 \58 \59 \5A \5B \5C \5D \5E \5F
\60 \61 \62 \63 \64 \65 \66 \67 \68 \69 \6A \6B \6C \6D \6E \6F
\70 \71 \72 \73 \74 \75 \76 \77 \78 \79 \7A \7B \7C \7D \7E \7F
\80 \81 \82 \83 \84 \85 \86 \87 \88 \89 \8A \8B \8C \8D \8E \8F
\90 \91 \92 \93 \94 \95 \96 \97 \98 \99 \9A \9B \9C \9D \9E \9F
\A0 \A1 \A2 \A3 \A4 \A5 \A6 \A7 \A8 \A9 \AA \AB \AC \AD \AE \AF
\B0 \B1 \B2 \B3 \B4 \B5 \B6 \B7 \B8 \B9 \BA \BB \BC \BD \BE \BF
\C0 \C1 \C2 \C3 \C4 \C5 \C6 \C7 \C8 \C9 \CA \CB \CC \CD \CE \CF
\D0 \D1 \D2 \D3 \D4 \D5 \D6 \D7 \D8 \D9 \DA \DB \DC \DD \DE \DF
\E0 \E1 \E2 \E3 \E4 \E5 \E6 \E7 \E8 \E9 \EA \EB \EC \ED \EE \EF
\F0 \F1 \F2 \F3 \F4 \F5 \F6 \F7 \F8 \F9 \FA \FB \FC \FD \FE \FF
```

## #translate

### Description

This directive tells SQLBase how to translate characters or strings of characters when sorting character strings or when evaluating strings for equality. You only need to specify characters which need to be translated.

### Example

If the sorting sequence in the country Glarpia requires:

- “N” to be translated to “Z”
- “W” to be translated to “M”
- “EF” to be translated to “F”
- “O” to be translated to “OH”

Then the *translate* directive would be:

```
[Glarpia]
#translate
N Z
W M
EF F
O OH
```

## #upper

### Description

Some countries require special handling when converting a string to uppercase. SQLBase automatically supports standard English uppercase conversion. The upper directive alters the uppercase conversion to match the needs of a country.

### Example

If the country Glarpia needs to convert “n” to “X” and “m” to “Y” when doing an uppercase operation, the upper directive would be:

```
[Glarpia]
#upper
n X
m Y
```

## #whitespace

### Description

This directive specifies characters that are white space. A white space character is ignored during parsing.

Follow this directive with a list of characters.

### Example

The country Japan would have this whitespace directive:

```
[Japan]
#whitespace
\9 \A \D \20
```

## Building a database with NLS

In order to use National Language Support, you must build your database from scratch, using country-specific items. You cannot change from one configuration to another without unloading your database and doing these steps:

1. Edit country.sql as appropriate for the language.
2. Copy country.dbs to the database directory.

```
md c:\sqlbase\germany
copy country.dbs c:\sqlbase\germany\germany.dbs
```

3. Put the country.sql file in the same directory as message.sql on BOTH the client and server.

4. Edit SQL.INI on BOTH the client and server, adding the country keyword. A server section would look like this:

```
[dbntrsv]
...
country=germany
dbname=germany,sqlws32
```

A client section would look like this:

```
[win32client]
...
country=germany
dbname=germany,sqlws32
...
; add next section if it does not exist
[sqltalk]
country=germany
```

5. Start SQLTalk and connect to the database. The first time you connect to a database that is a copy of country.dbs, SQLBase initializes it.
6. Run the country.tlk script file.  

```
run country.tlk
```

This script creates views for the database.
7. Make a copy of the new database, naming it start.dbs. Use this start.dbs to create new databases in the language.

---

**Important:** You cannot mix and match different country databases on the same server. The server expects all databases to support the same language.

---

---

**Note:** For partitioned databases, make sure you first build the *start.dbs* file with your own *country.sql* file. Then create your partitioned database.

---

## Chapter 11

# SQLBase and NetWare Directory Services(NDS)

---

NDS (NetWare Directory Services) provides global access to all network resources, regardless of their physical location by providing a single, network wide directory. This directory is accessible from multiple points by network users, services, and applications. NDS also includes bindery emulation which provides compatibility with bindery-based versions of applications, utilities, and services that can coexist with NDS on the network.

This chapter describes how to configure SQLBase to use NDS 4.1.1 and higher, including:

- Configuring the server
- Setting up the SQLBase NDS classes
- Using bindery emulation
- Configuring clients

## SQLBase and NDS

NetWare 4.11 and higher include NetWare Directory Services (NDS), which replaces the NetWare 3.x Bindery database. SQLBase server for NetWare supports NDS. With NDS, you can advertise SQLBase servers and databases on the network in the directory tree. SQLBase server for NetWare provides a set of SQLBase classes and attributes used to create SQLBase objects in the NDS tree.

---

**Note:** This chapter assumes you are familiar with NDS and how to use it. For details and background on the NDS, read Novell's NDS documentation.

---

All NDS clients, except those using Novell's DOS Requester (also known as NETX) or VLM, are able to see objects in the NDS tree.

## Configuring SQLBase Server for NetWare for NDS

When you installed SQLBase for NetWare, you entered information in a series of dialogs. The install program used this information to configure SQL.INI. This section explains how the install program configured SQL.INI, including setting the:

- Insertion context
- NDS login ID and password
- Advertising mode

### Insertion context

The NDS directory tree is an hierarchical structure that holds and organizes objects. An object's location in the tree is referred to as its context. The `insertioncontext` keyword specifies where in the NDS directory tree the SQLBase server object resides.

The syntax for this keyword is:

```
INSERTIONCONTEXT = object_name
```

The *object\_name* can be a complete or partial object name, containing the following standard abbreviations, which are the most commonly used name types:

| Object type         | Abbreviation |
|---------------------|--------------|
| Country             | C            |
| Organization Name   | O            |
| Organizational Unit | OU           |
| Locality Name       | L            |
| Common Name         | CN           |

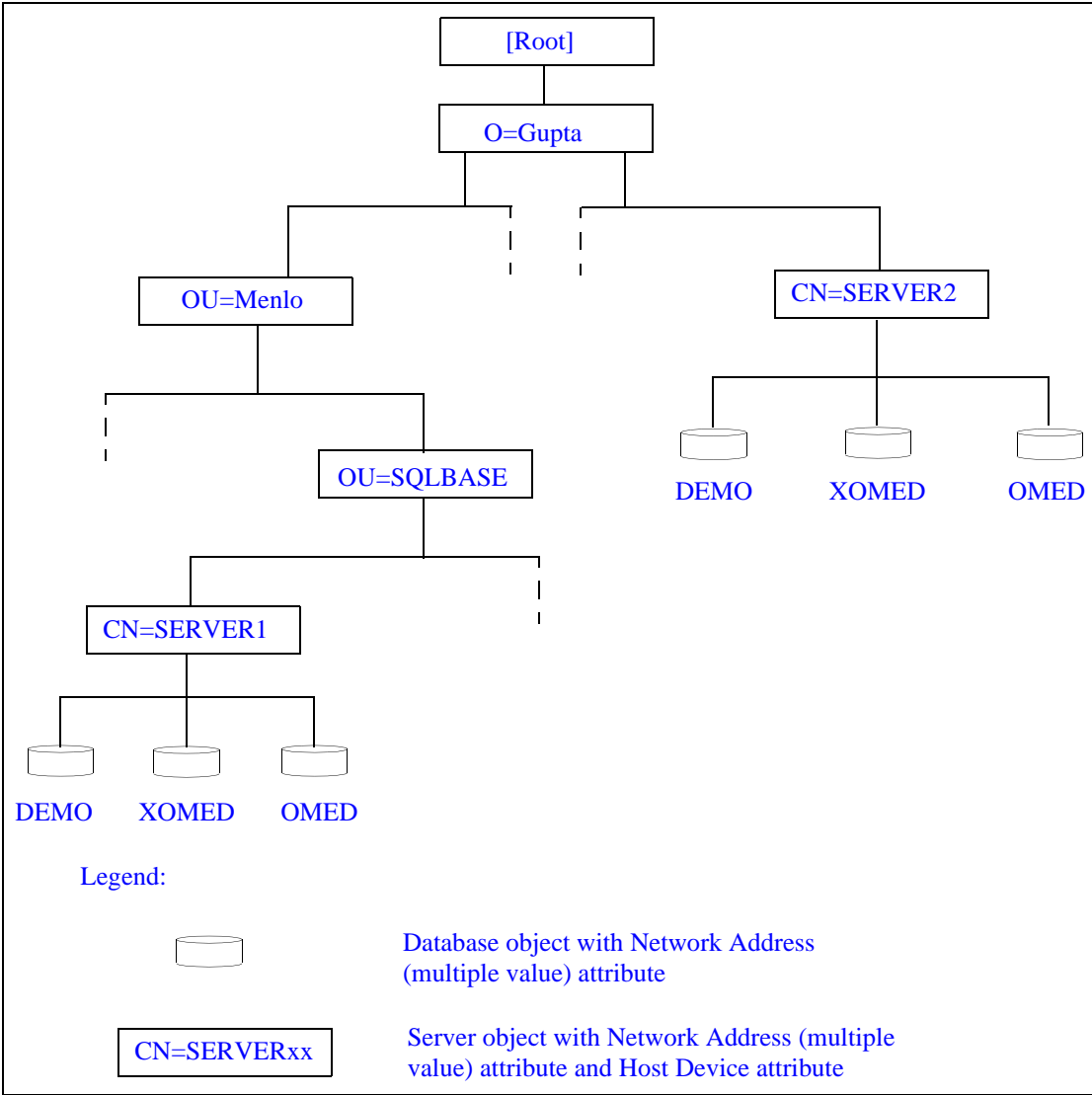
If you use a complete object name you provide the path from the root of the NDS tree to the server object.

The server context is specified as a list of containers, separated by periods, between the root of the tree and the server. For example, if you want to provide an insertion context in the SQLBASE organizational unit:

```
[dbnwsrv]
insertioncontext=OU=SQLBASE.OU=Menlo.O=Gupta
```

or, if you want the server to reside in the Gupta organizational unit of the NDS tree, you enter:

```
[dbnwsrv]
insertioncontext=O=Gupta
```



**Note:** Server objects and database objects are instances of the classes `SQLBASE::DBServer` and `SQLBASE::Database` respectively. For more, read *SQLBase NDS schema extension* on page 11-6.

---

**Important:** Be sure each *searchcontext* keyword in the client sections of SQL.INI is set to the same value as the *insertioncontext* keyword for the server. For details on search context, read *Insertion context* on page 11-2.

---

The *insertioncontext* keyword is optional. If you do not specify this keyword, the default context of the user specified by the *ndsloginid* keyword.

## NDS logon ID and password

The *ndsloginid* and *ndsloginpassword* keywords specify the logon ID and password used to log into the NDS directory tree for adding, deleting, and modifying the SQLBase server and database classes.

The syntax for these keywords is:

```
ndsloginid=userid
ndsloginpassword=password
```

For example:

```
[dbnwsrv]
ndsloginid=admin
ndsloginpassword=dev
```

These keywords are mandatory only if NDS is used for advertising SQLBase servers and databases. Otherwise these keywords are ignored.

## Advertising mode

In order to provide backward compatibility, NetWare 4.1.x and later provides the option of bindery emulation using SAP (Service Advertising Protocol). This lets NetWare clients that find network resources through the bindery connect to NetWare 4.1.x and later servers. For more, read *Setting bindery emulation* on page 11-9.

You can use bindery emulation alone or in conjunction with NDS by setting the advertising mode for SQLBase servers and databases with the *nwadvertisemode* keyword.

If you use NDS with bindery emulation, SAP advertises the presence of databases and servers on the network. If you have clients relying on SAP location information, you can configure *nwadvertisemode* to allow both SAP and NDS for advertising on the network. If you do not need to use SAP, you can use this keyword to turn it off which avoids the additional network traffic.

The *nwadvertisemode* keyword is mandatory if NDS or both NDS and SAP are used for advertising SQLBase servers and databases. Otherwise the default for this keyword is SAP only.

The syntax for this keyword is:

```
nwadvertisemode = 1 | 2 | 3
```

The meaning of the parameter is:

- 1 NDS only
- 2 SAP via bindery emulation only (default)
- 3 Both SAP (via bindery emulation) and NDS

For example, this specifies to use NDS only:

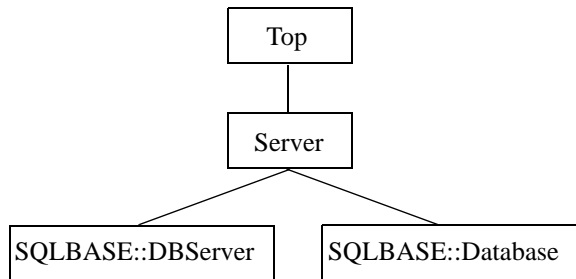
```
nwadvertisemode=2
```

## SQLBase NDS schema extension

SQLBase provides a schema extension utility called **ndsschma.nlm** that you run at the server console. This utility defines these two SQLBase-specific classes to NDS:

- SQLBASE::DBServer
- SQLBASE::Database

Both of these classes are derived from the built-in NDS Server class.



The next section presents information about these classes.

When you run **ndsschma.nlm**, you are prompted for a user name and password. This user must have create and delete rights in the NDS tree.

## SQLBase classes

This section first presents general information defined for each of the NDS classes before describing the SQLBase classes. Read this section first to clarify the SQLBase class descriptions that follow.

Each NDS class defines the following:

| Object Class Information          | Description                                                                                                                                                                                                                                                                                                                                                          |
|-----------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Type                              | “Effective” means that objects can be instances of this class; “non-effective” means that this class can only be derived by other classes.                                                                                                                                                                                                                           |
| Super classes                     | The base classes from which this class is derived.                                                                                                                                                                                                                                                                                                                   |
| Containment                       | The NDS container in which objects of this class can reside.                                                                                                                                                                                                                                                                                                         |
| Named By                          | The partial name or Relative Distinguished Name (RDN) of objects of the class consists of at least one of the attributes listed here. At least one of the attributes listed here must be given a value when creating an object of the class. Attributes listed in this section are also listed in the Mandatory Attributes or Optional Attributes sections.          |
| Mandatory and Optional Attributes | All attributes are either mandatory or optional. If an attribute is mandatory, a value must be assigned to that attribute. If an attribute is optional, an assigned value is not required unless it is the only Named By Attribute.                                                                                                                                  |
| Default ACL Template              | Every expanded class definition has an ACL attribute (inherited from the class Top). This attribute holds information about which trustees have access to the object itself and which trustees have access to the attributes for the object. Each class can have its own default set of values for their ACL but can also inherit ACL values from the super classes. |

---

**Note:** Every attribute marked with an \* is an inherited attribute from its super class.

---

The SQLBase Server and SQLBase database classes have SQLBASE:: as prefix because it is registered as the requested name prefix for the SQLBase product with Novell’s Directory Schema Registry.

## SQLBASE::DBServer

This class identifies entities that manage SQLBase servers.

### Type

Effective

### Super classes

Server

### Containment classes

\*Top

\*Country

\*Locality

\*Organization

\*Organizational Unit

### Named By

\*CN (Common Name)

### Mandatory Attributes

\*Host Device

### Optional Attributes

\*Network Address

### Default ACL Template

*Object NameDefault RightsAffected Attributes*

\*[Creator]Supervisor[Entry Rights]

## SQLBASE::Database

This class identifies entities that manage SQLBase databases.

### Type

Effective

### Super Classes

Server

## Containment Classes

SQLBASE::DBServer

## Named By

None

## Mandatory Attributes

None

## Optional Attributes

\*Network Address

## Default ACL Template

*Object NameDefault RightsAffected Attributes*

\*[Creator]Supervisor[Entry Rights]

# Setting bindery emulation

With bindery emulation, NDS imitates a flat structure for leaf objects within an Organization or Organizational Unit object. When bindery emulation is enabled, all objects within the specified container can be accessed by both NDS clients and binary-based clients. The bindery emulator retrieves information stored in a bindery portion of the tree on a NetWare server and sends bindery-like information to bindery clients. Bindery emulation applies only to leaf objects in the specified container object.

The container(s) where bindery services is set is called the bindery context. You set the bindery context by giving the SET BINDERY CONTEXT command on a NetWare server. For example:

```
SET BINDERY CONTEXT = object_name
```

You can set up to 16 bindery contexts per server

The *object\_name* can be a complete or partial object name, containing the standard names used in the NDS tree, separated by periods. For examples, read *Insertion context* on page 11-2.

The SET BINDERY CONTEXT command starts full bindery emulation which enables bindery clients to log into the NetWare server and clients can have the server be their preferred server. If you do not set the bindery context, clients cannot log into or get information from the NDS tree.

Full bindery emulation means that the emulator is in a *dynamic* level and *static* level:

- The dynamic level is for objects that advertise and are automatically deleted when they shut down such as servers and printers
- The static level is for objects created permanently until someone deletes them such as users, groups, and queues

This provides full-scale emulation of static objects as well as accepting requests for dynamic objects.

When the bindery context is not set (by giving the command “SET BINDERY CONTEXT=” without a parameter), the bindery emulation is in a *dynamic only* level, which means the emulation only accepts requests for dynamic objects.

For more about the bindery emulator, read Novell’s documentation.

## Configuring Clients for NDS

You can provide optional configuration settings to determine how a NDS client searches for objects when a client logs into a NetWare server:

- The search order if both NDS and bindery emulation is used on your network
- The search context used to search the directory tree if NDS is used on your network

## NDS logon ID

Be sure that all clients that use NDS have an NDS logon ID. When logging into the system, the primary logon ID provided by the client must be one used exclusively for NDS.

## Search order

If you use NDS with bindery emulation, you can specify the search order that NDS clients use to connect to SQLBase. You can specify if the client searches the NDS tree before the bindery, or vice versa. For more, read *Setting bindery emulation* on page 11-9.

The *preferrednameservice* keyword specifies whether NDS is searched before or after the bindery database when an NDS client logs into the server.

The syntax for this keyword is:

```
preferrednameservice = NDS | BIN
```

The meaning of the parameter is:

|     |                                             |
|-----|---------------------------------------------|
| NDS | The NDS tree is searched before the bindery |
| BIN | The bindery is searched before the NDS tree |

This keyword is optional. The default is NDS when an NDS client logs into the server. If the client is non-NDS, the keyword is ignored and only the bindery is searched.

For example, to make a client search NDS before the Bindery, in the client section ([win32client]) of SQL.INI specify:

```
[win32client.spx32]
preferrednameservice=NDS
```

## Search context

The searchcontext keyword specifies the part of the NDS directory tree where an NDS client searches for the SQLBase server names and installed database names.

You can specify more than one searchcontext keyword. If you do specify two or more keywords, the search begins from the first to the last search context that you specify, and if the object is not found, the current context of the logged in user is searched. Once an object is resolved, the search ends at that particular context.

---

**Note:** Be sure the *searchcontext* keyword is set to the same value as the *insertioncontext* keyword in the [server] section of SQL.INI.

---

The syntax for this keyword is:

```
searchcontext = object_name
```

The *object\_name* can be a complete or partial object name, containing the standard names used in the NDS directory tree, separated by periods. For examples, read *Insertion context* on page 11-2.

To specify that clients search a part of the directory tree identified as O=Gupta:

```
[win32client.spx32]
searchcontext=O=Gupta
```

The *searchcontext* keyword is optional. If this keyword is not specified, the entire NDS directory tree is searched.

If the client is non-NDS, the keyword is ignored.



## Chapter 12

# Running SQLBase Server as a Windows Service

---

This chapter contains information on running SQLBase Server as an application program or a Windows service program. Topics include:

- Overview of service programs
- Installing SQLBase Server for Windows as a service program
- Administering service programs
- Services window of the Control Panel
- Using the Windows registry
- Windows Event Log
- Shutting down SQLBase Server for Windows when running as a service program

## About service programs

SQLBase Server runs as a service in Windows NT (4.0 and later), 2000, XP, and Server 2003. Throughout this chapter, the term "Windows" will refer to those three operating systems.

Windows programs can be either application programs or service programs. In earlier versions of SQLBase, the server ran as an application program, requiring the user to log onto Windows before starting SQLBase. This results in SQLBase terminating when the user logs off.

Windows service programs run independently of the user, and are unaffected by users logging on and off the Windows machine. They do not require the user to log onto the Windows machine prior to starting, and can start either automatically or on demand. If started automatically, they begin their execution during Windows initialization.

Running SQLBase Server as a Windows service program offers the following advantages:

- A user need not be logged on to the system for SQLBase to start.
- Security is enhanced, preventing unauthorized tampering or termination.

You can start SQLBase as a Service manually or automatically. When you specify manual start up, you must start Windows before the SQLBase Service program starts up. When you specify automatic start up, the SQLBase Service program starts up automatically when Windows starts up.

You can administer the SQLBase Windows service program using NT Service Manager, the Services tool in Windows 2000 or Windows XP or Server 2003, or SQLBase Management Console (SMC). For information on SMC, read on page 5-26.

## Installing SQLBase Server for Windows

In order to install SQLBase Server for Windows as a service program, the user performing the installation must have Administrator privileges.

SQLBase Server for Windows always installs as a service program. However, it can also be run as an application program. For more, read *Administering service programs* on page 12-3.

The default startup setting is with SQLBase enabled as a service. During installation, SQLBase is set to run as a system account and to interact with the desktop.

## Administering service programs

You can administer SQLBase as a service on Windows NT (4.0 and later), 2000, XP, and Server 2003 using one or both of the following tools:

- Gupta's SQLBase Management Console (SMC), which is part of the SQLBase Server package
- Services window of the Control Panel

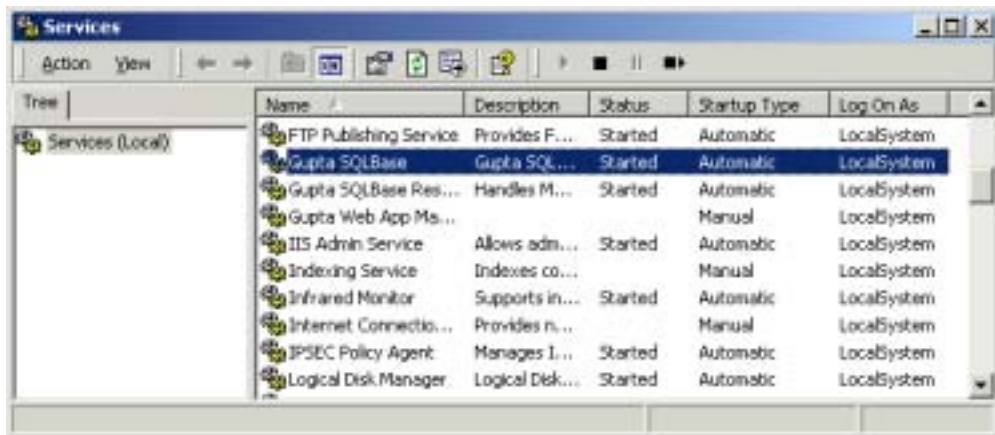
---

**Note:** SQLBase Management Console provides authorization to any user to administer SQLBase Server if they have access to the system running SQLBase. If you want to restrict access to SQLBase Server to a single authorized user or administrator, you must administer SQLBase as an service exclusively through the Windows Services window.

---

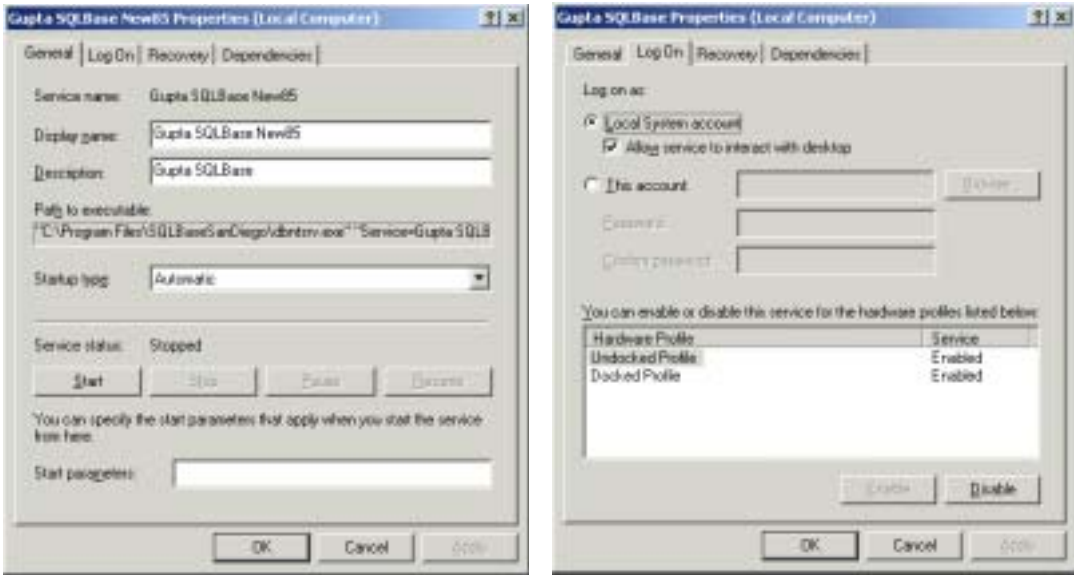
## Using the Services window of the Control Panel

You can administer SQLBase as an Service program using the Services window of the Control Panel. The Services window (Windows 2000 is shown here) displays the current status and startup type of each service program installed on the machine. Use the Start and Stop buttons to start and stop the selected service program.



*Windows 2000 Services window*

The Properties toolbar button displays a tabbed dialog.



### *Properties dialog*

Use this window to modify the properties of the service program, including startup type and the account under which the service program executes. If SQLBase Server for Windows is running as a service program, and is running in an account other than the system account, the normal SQLBase interface will be completely hidden. This limitation is imposed by Windows. In this situation, administrative access, beyond starting and stopping SQLBase, is gained through SMC. If the service program logs on as a System Account, it can be set to interact with the desktop, but must be stopped using SMC or Windows Services Manager.

## Service information in the Windows Registry

SQLBase has an entry in the Windows registry that identifies basic parameters for the Windows Service Control Manager and SQLBase Management Console. The registry key for this entry is:

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\
 Gupta SQLBase\servername
```

The term *servername* refers to the name you gave to the server during SQLBase installation, or later by modifying the server name using SMC. There may be multiple registry entries for SQLBase 8.5 servers, each ending in a different server name. There may also be a similar registry entry without a server name - this would indicate a SQLBase server of a version earlier than 8.5.



### *Windows services registry key for SQLBase*

The SQLBase install program initially makes this entry. The values may be modified later by changes you make through SQLBase Management Console or Connectivity Administrator.

The ImagePath entry includes an entire command line: The path and name of the server executable is followed by a command-line argument with the path and name of the configuration file to be used. The default name is SQL.INI and the default path is the same as the server executable, but you can change those values in Connectivity Administrator or SQLBase Management Console.

SQLBase stores two types of configuration information in the registry:

- Configuration information that is used by SQLBase client applications.
- Service configuration information used only when running as a service (described here).

## Windows Application Event Log

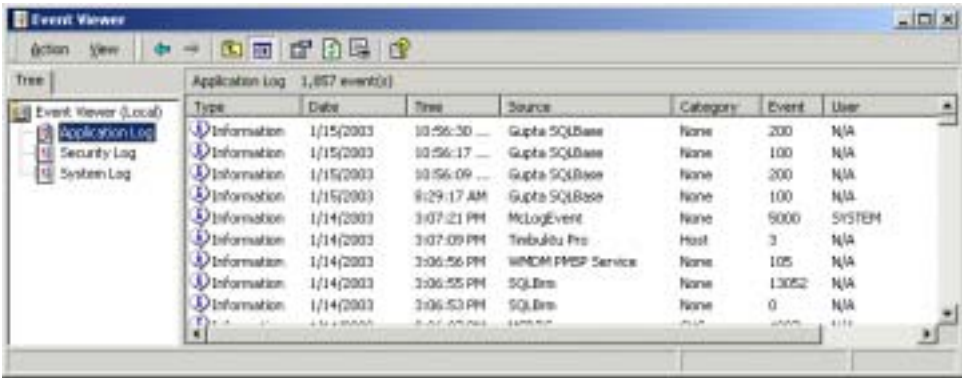
When running SQLBase Server for Windows as a service, operational and error messages are logged to the Windows Application Event Log. Message types include:

- Startup
- Shutdown
- Database(s) being serviced
- SQL.INI file errors
- Internal logic errors
- Deadlock messages

**Note:** SQLBase displays no error dialogs when running as a service program, even if configured to interact with the desktop.

The Windows Application Event Log supports three types of logging events:

- Informational events
- Warning events
- Error events



*Windows Event Log*

Click on an event to bring up the **Event Detail** dialog.



*Event Detail dialog message text*

A parameter in the Windows registry controls event logging. The following table lists the SQLBase level, event type, and message events:

| SQLBase Level | Windows Event Type                                                                                                                  | SQLBase Message Events                                                                                                                                                                                                                                       |
|---------------|-------------------------------------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 1             | <ul style="list-style-type: none"><li>• Informational</li><li>• Error</li><li>• Error</li><li>• Warning</li><li>• Warning</li></ul> | <ul style="list-style-type: none"><li>• Startup and Shutdown</li><li>• Startup errors (such as SQL.INI errors)</li><li>• Database shutdown due to error conditions</li><li>• Deadlock detection</li><li>• Shutting down with users still connected</li></ul> |

Each SQLBase message event has corresponding message text that displays in the SQLBase GUI when the event occurs. The same text is recorded in the Windows Event Log.

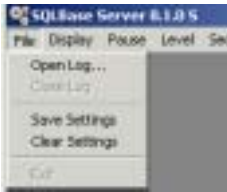
You can also view errors using the Windows Event Viewer.

## Shutting down SQLBase Server for Windows

When you are running SQLBase Server for Windows as an application program, SQLBase can be shut down using SQLBase Management Console, the SQLBase File menu, or the SQLBase icon when the user interface is minimized.

When you are running SQLBase Server for Windows as a service, the ability to shut down SQLBase from a menu item is disabled. You can only shut down SQLBase using SQLBase Management Console or from the Services window of the Control Panel.

If users are connected to SQLBase, a database recovery is required on the next start up. To prevent data loss, be sure all users are logged off SQLBase before shutting it down. .



*Disabled File menu shutdown point*

## Running SQLBase as an application program

SQLBase Server for Windows may still be set up to run as an application program, even if it is installed as a service. You can only run one instance of SQLBase on a

machine. If SQLBase is already running as a service, then trying to run SQLBase as an application program will be unsuccessful and result in an error message from the new server. When choosing to run SQLBase as an application program, you must establish a Windows session before you start the SQLBase Server. To start SQLBase as an application program, double-click on the SQLBase Server icon rather than starting SQLBase with the Windows Service Manager or the SQLBase Management Console.

You can shut down SQLBase using SQLBase Management Console, SQLBase File menu, or the SQLBase icon when the user interface is minimized. If the SQLBase Server is running when you end a Windows session, the SQLBase Server is terminated. When running SQLBase Server as an application program, you can use the SQLBase Management Console to report the status of the SQLBase databases.



## Appendix A

# System Catalog Tables

---

This Appendix contains descriptions of the tables in SQLBase's system catalog. The tables are organized alphabetically by name.

# System catalog summary

The system catalog is a set of tables that contain information about objects in the database. The tables are owned by SYSADM and maintained by SQLBase. The system catalog is also called the *data dictionary*.

| Table Name        | Description                                                                               |
|-------------------|-------------------------------------------------------------------------------------------|
| SYSCOLAUTH        | Lists each user's column update privileges.                                               |
| SYSCOLUMNS        | Lists each column of every table.                                                         |
| SYSCOMMANDS       | Lists all stored commands and procedures.                                                 |
| SYSDEPENDENCIES   | Lists each dependency between a stored procedure and an external function.                |
| SYSEVENTS         | Lists all the system timer events.                                                        |
| SYSEXECUTEAUTH    | Lists the execution authority levels of users with privileges on each stored procedure.   |
| SYSEXTFUN         | Lists all declared external functions.                                                    |
| SYSEXTPARAM       | Lists each parameter of an external function.                                             |
| SYSFKCONSTRAINTS  | Lists each foreign key constraint.                                                        |
| SYSINDEXES        | Lists each table's indexes.                                                               |
| SYSOBAUTH         | Lists each user who is granted execute privilege on an external function.                 |
| SYSOBSYN          | Lists each synonym created for an external function, stored command, or stored procedure. |
| SYSKEYS           | Lists each column in every index.                                                         |
| SYSPARTTRANS      | Lists each in-doubt distributed transaction.                                              |
| SYSPKCONSTRAINTS  | Lists each primary key constraint.                                                        |
| SYSROWIDLISTS     | Lists information about saved Result sets.                                                |
| SYSSYNONYMS       | Lists all table and view synonyms.                                                        |
| SYSTABAUTH        | Lists each user's table privileges.                                                       |
| SYSTABCONSTRAINTS | Lists each table constraint.                                                              |
| SYSTABLES         | Lists each table or view.                                                                 |

| Table Name  | Description                                          |
|-------------|------------------------------------------------------|
| SYSTRGCOLS  | Lists each column on which an UPDATE trigger exists. |
| SYSTRIGGERS | Lists each trigger.                                  |
| SYSUSERAUTH | Lists each user's database authority level.          |
| SYSVIEWS    | Lists the text of each view.                         |

## SYSADM.SYSCOLAUTH

This table contains the update privileges of users for individual columns of a table or view.

| Column Name | Description                                                                                     |
|-------------|-------------------------------------------------------------------------------------------------|
| GRANTEE     | The authorization ID of the user who holds update privileges.                                   |
| CREATOR     | The authorization ID of the user who created the table on which the update privileges are held. |
| TNAME       | The name of the table or view on which privileges are held.                                     |
| COLNAME     | The name of the column to which the UPDATE privilege applies.                                   |

## SYSADM.SYSCOLUMNS

This table contains one row for every column of each table and view (including the columns of the system catalog tables).

| Column Name | Description                                                                                           |
|-------------|-------------------------------------------------------------------------------------------------------|
| NAME        | The name of the column.                                                                               |
| TBNAME      | The name of the table or view that contains this column.                                              |
| TBCREATOR   | The authorization ID of the user who created the table or view in which the column exists.            |
| COLNO       | The relative column number in the table. This number remains the same even after a column is dropped. |
| COLTYPE     | The data type of the column.                                                                          |

| Column Name  | Description                                                                                                                                                                                                                                                                                                                                                                                                                    |         |   |          |   |       |   |      |                  |         |                          |         |                     |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|---------|---|----------|---|-------|---|------|------------------|---------|--------------------------|---------|---------------------|
| LENGTH       | <p>The length of the data in the column, or for a DECIMAL column, the precision.</p> <p>The standard lengths for the data types are:</p> <table><tr><td>INTEGER</td><td>4</td></tr><tr><td>SMALLINT</td><td>2</td></tr><tr><td>FLOAT</td><td>8</td></tr><tr><td>CHAR</td><td>Length of string</td></tr><tr><td>VARCHAR</td><td>Maximum length of string</td></tr><tr><td>DECIMAL</td><td>Precision of number</td></tr></table> | INTEGER | 4 | SMALLINT | 2 | FLOAT | 8 | CHAR | Length of string | VARCHAR | Maximum length of string | DECIMAL | Precision of number |
| INTEGER      | 4                                                                                                                                                                                                                                                                                                                                                                                                                              |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| SMALLINT     | 2                                                                                                                                                                                                                                                                                                                                                                                                                              |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| FLOAT        | 8                                                                                                                                                                                                                                                                                                                                                                                                                              |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| CHAR         | Length of string                                                                                                                                                                                                                                                                                                                                                                                                               |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| VARCHAR      | Maximum length of string                                                                                                                                                                                                                                                                                                                                                                                                       |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| DECIMAL      | Precision of number                                                                                                                                                                                                                                                                                                                                                                                                            |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| SCALE        | <p>The scale for a DECIMAL data type column. This is zero (0) for other types.</p>                                                                                                                                                                                                                                                                                                                                             |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| NULLS        | <p>'Y' if nulls are allowed in the column; 'N' if defined with NOT NULL; 'D' if defined with NOT NULL WITH DEFAULT.</p>                                                                                                                                                                                                                                                                                                        |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| UPDATES      | <p>'Y' if the column can be updated; 'N' if the column is read-only.</p>                                                                                                                                                                                                                                                                                                                                                       |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| REMARKS      | <p>A user-specified comment about each column. The maximum length of a comment is 254 characters.</p>                                                                                                                                                                                                                                                                                                                          |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| LABEL        | <p>A user-specified label about each row. The maximum length of a label is 30 characters.</p>                                                                                                                                                                                                                                                                                                                                  |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| AVGCOLLEN    | <p>The average column length across all rows in the table. This can differ from the defined column length because SQLBase stores all columns as variable data.</p>                                                                                                                                                                                                                                                             |         |   |          |   |       |   |      |                  |         |                          |         |                     |
| AGCOLLONGLEN | <p>The average length of a LONGVARCHAR column across all rows of the table. This is zero for a non-LONGVARCHAR column.</p>                                                                                                                                                                                                                                                                                                     |         |   |          |   |       |   |      |                  |         |                          |         |                     |

## SYSADM.SYSCOMMANDS

This table contains one row for every stored command or procedure.

| Column Name   | Description                                                                                                                                                                                                                                                                       |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREATOR       | The authorization-id of the creator of the stored command or procedure.                                                                                                                                                                                                           |
| NAME          | The name of the stored command or procedure.<br><br>When SQLBase stores a command contained in a static procedure, SQLBase assigns it an internally-generated name.                                                                                                               |
| TYPE          | Command (C) or procedure (P).                                                                                                                                                                                                                                                     |
| SYSTEM        | Contains 'Y' if SQLBase created the stored command or procedure; 'N' if a user created it.<br><br>When SQLBase stores a command or procedure contained in a static procedure, this value is 'Y'. Conversely, when you explicitly STORE a command or procedure, this value is 'N'. |
| STATIC        | 'Y' if the stored procedure is static; 'N' if it is not.                                                                                                                                                                                                                          |
| VALID         | Contains 'Y' if the command or procedure is valid, and 'N' if it is invalid.<br><br>A stored command or procedure can become invalid in a situation such as when you delete a referenced column, or drop an index on a referenced column.                                         |
| AUTORECOMPILE | If set to ON and the stored command later becomes invalid, SQLBase automatically recompiles the stored command the next time you execute it with either the SQLTalk EXECUTE command or the SQL/API <i>sqlret</i> function.                                                        |
| TEXT          | The text of the stored command or SQL statements in the procedure.                                                                                                                                                                                                                |
| SNUM          | An integer column. This is an identification (serial number) associated with SYSCOMMANDS.                                                                                                                                                                                         |

SYSADM.SYSDEPENDENCIES

This table currently contains one row for each dependency between a stored procedure and an external function.

| Column Name | Description                                                                                                                       |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------|
| DETCREATOR  | The authorization ID of the user who created the determinant object, which is the external function.                              |
| DETNAME     | The name of the determinant object, which is the external function. This is the name assigned to the function when it is created. |
| DETTYPE     | The type of determinant object. 'F' for function.                                                                                 |
| DEPCREATOR  | The authorization ID of the user who created the dependent object, which is the stored procedure.                                 |
| DEPNAME     | The stored procedure name which is the dependent object of the external function.                                                 |
| DEPTYPE     | The type of dependent object. 'P' for procedure.                                                                                  |

SYSADM.SYSEVENTS

This table contains one row for every event.

| Column Name | Description                                                                                                                    |
|-------------|--------------------------------------------------------------------------------------------------------------------------------|
| CREATOR     | The authorization ID of the creator of the event.                                                                              |
| NAME        | The name of the event.                                                                                                         |
| TYPE        | Command (C) or procedure (P).                                                                                                  |
| TYPEID      | The event type. Currently, only the timer event type is supported.                                                             |
| EVENTID     | The ID of the event.                                                                                                           |
| BEGINTIME   | The start time of the event.                                                                                                   |
| INTERVAL    | The time interval (in seconds) between executions of the event.                                                                |
| SYSTEM      | 'Y' if the event is system-defined; 'N' if it is not.                                                                          |
| SPSNUM      | An integer column that refers to the serial number (SNUM in SYSCOMMANDS) of the stored procedure that is executed by an event. |

| Column Name | Description                                                                                                |
|-------------|------------------------------------------------------------------------------------------------------------|
| TEXT        | The text of the stored command or SQL statements in the procedure.                                         |
| ACTIONCODE  | The compiled code of the stored command or SQL statements in the procedure. This is for internal use only. |

## SYSADM.SYSEXECUTEAUTH

This table contains one row for every GRANT PRIVLEGES command run for stored procedures.

| Column Name    | Description                                                            |
|----------------|------------------------------------------------------------------------|
| CREATOR        | The authorization ID of the user who created the stored procedure      |
| NAME           | An internally-generated name for the stored procedure.                 |
| GRANTEE        | The authorization ID of the grantee (user of the stored procedure).    |
| USECREATORPRIV | 'Y' if the GRANTEE is using the creator's privileges; blank otherwise. |
| USEGRANTEEPRIV | 'Y' if the GRANTEE is using his own privileges; blank otherwise.       |

## SYSADM.SYSEXTFUN

This table contains one row for each declared external function.

| Column Name | Description                                                         |
|-------------|---------------------------------------------------------------------|
| CREATOR     | The authorization ID of the user who created the external function. |
| NAME        | The name of the external function.                                  |
| EXTNAME     | The external name of the external function.                         |
| LIBNAME     | Full path name of the library where the external function resides   |
| SNUM        | Unique SQLBase ID for the external function.                        |
| NUMPARAMS   | Number of parameters in the external function.                      |

| Column Name | Description                                                                                                                                                     |
|-------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------|
| RETURN      | ‘Y’ if the external function returns a value; ‘N’ otherwise.                                                                                                    |
| EXECMODE    | Function execution mode: ‘S’ for same thread, or ‘P’ for separate process.                                                                                      |
| CALLSTYLE   | Callstyle for the external function. It can be one of the following: <ul style="list-style-type: none"><li>• CDECL</li><li>• STDCALL</li><li>• PASCAL</li></ul> |
| REMARKS     | User-specified comment                                                                                                                                          |
| LABEL       | User-specified label.                                                                                                                                           |

SYSADM.SYSEXTPARAM

This table contains one row for each parameter of an external function.

| Column Name | Description                                                      |
|-------------|------------------------------------------------------------------|
| FUNSNUM     | Unique SQLBase ID for the external function.                     |
| POSITION    | Position of the parameter as defined in the function (1, 2 ...). |
| EXTTYP      | External data type for the parameter.                            |
| RECEIVE     | ‘Y’ if the external function returns a value; ‘N’ otherwise.     |

SYSADM.SYSFKCONSTRAINTS

This table contains each table’s foreign key columns.

| Column Name | Description                                                           |
|-------------|-----------------------------------------------------------------------|
| CREATOR     | The authorization ID of the user who created the table.               |
| NAME        | The name of the table to which foreign key applies.                   |
| CONSTRAINT  | The name of the foreign key constraint.                               |
| FKCOLSEQNUM | The sequence number of the foreign key column within the foreign key. |

| Column Name   | Description                                                                                  |
|---------------|----------------------------------------------------------------------------------------------|
| REFSCOLUMN    | The name of the column in the dependent table which has the foreign key.                     |
| REFDTBCREATOR | The authorization ID of the user who created the parent table referenced by the foreign key. |
| REFDTBNAME    | The name of the parent table referenced by the foreign key.                                  |
| REFDCOLUMN    | The column name of the referenced column in the parent table.                                |

## SYSADM.SYSINDEXES

This table contains one row for every index, including indexes on system catalog tables.

| Column Name | Description                                                                                                                                    |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------|
| NAME        | The name of the index.                                                                                                                         |
| CREATOR     | The authorization ID of the creator of the index.                                                                                              |
| TBNAME      | The name of the table on which the index is defined.                                                                                           |
| TBCREATOR   | The authorization ID of the creator of the table on which the index is defined.                                                                |
| UNIQUERULE  | Indicates whether the index is unique: 'D' if duplicates are allowed and 'U' if the index is unique.                                           |
| COLCOUNT    | The number of columns in the index.                                                                                                            |
| IXTYPE      | Indicates the type of index: 'H' if hashed and 'B' if B+tree.                                                                                  |
| CLUSTERRULE | Indicates whether the index is clustered: 'Y' if clustered and 'N' if not.                                                                     |
| SYSTEM      | 'Y' if the index is system-defined; 'N' if it is not.                                                                                          |
| IXSIZE      | The number of rows in the table, as specified by the user.                                                                                     |
| PERCENTFREE | The amount of free space to leave on each index page when the index is first built. If not specified by the user, this defaults to 10 percent. |

| Column Name    | Description                                                                                                                                                                                                                                                                                                                                                                                                                                |
|----------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| HEIGHT         | <p>The height of the index. Minimum value is 1, which indicates that the index has only one page.</p> <p>The height of the index tree, also called depth, is the number of nodes that have to be read from the root to the leaf level, inclusive. This statistic is maintained dynamically in the index's control page, but is only recorded in the SYSINDEXES table either when you first create the index, or run UPDATE STATISTICS.</p> |
| INDEXPAGECOUNT | For a B+tree index, this is the number of index pages. For a hash index, this is null.                                                                                                                                                                                                                                                                                                                                                     |
| LEAFCOUNT      | For a B+tree index, this is the number of nodes in the bottom leaves of the index (leaf page). It is also the number of pages in the index's sequence set. For a hash index, this is null.                                                                                                                                                                                                                                                 |
| CLUSTERCOUNT   | <p>For a B+tree index, this is the cluster count, which is the total number of page changes that would occur if the entire table was read through the index's sequence set.</p> <p>The minimum value (for a perfectly clustered B+tree index) is the number of data pages. The maximum value (for a completely unclustered index) is the number of rows in the table. For a hash index, this is null.</p>                                  |
| PRIMPAGECOUNT  | For a B+tree index, this is null. For a hash index, this is the number of primary pages allocated for the index's subject table. This is the same as the number of hash slots available for distributing the table's row.                                                                                                                                                                                                                  |
| OVFLPAGECOUNT  | For a B+ tree index, this is null. For a hash index, this is the number of overflow pages allocated for the index's subject table. When you first create the table and index, this is the number of overflow pages that SQLBase pre-allocates. Subsequently, this number increases as additional overflow pages are required to handle hashing collisions.                                                                                 |
| AVGKEYLEN      | For a B+ tree index, this is the average key length across all index entries. This statistic is necessary because SQLBase maintains all index entries as variable length, minimum prefix only fields. For a hash index, this is null.                                                                                                                                                                                                      |
| GROUPNUM       | For a B+ tree index, this is the group number for index pages. For a hash index, this is the group number of overflow pages.                                                                                                                                                                                                                                                                                                               |

| Column Name | Description |
|-------------|-------------|
| USECOUNT    | Not in use. |

## SYSADM.SYSKEYS

This table contains one row for each column in an index.

| Column Name   | Description                                                                                                                                                                                                                                                                                                                                                            |
|---------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| IXNAME        | The name of the index.                                                                                                                                                                                                                                                                                                                                                 |
| IXCREATOR     | The authorization ID of the creator of the index.                                                                                                                                                                                                                                                                                                                      |
| COLNAME       | The name of the column of the key.                                                                                                                                                                                                                                                                                                                                     |
| COLNO         | The numerical position of the column in the row; for example 2 (out of 7 columns in the table).                                                                                                                                                                                                                                                                        |
| COLSEQ        | The numerical position of the column in the key; for example 2 (out of 3 columns constituting the key).                                                                                                                                                                                                                                                                |
| ORDERING      | The sort order of the column in the key. 'A' if ascending and 'D' if descending.                                                                                                                                                                                                                                                                                       |
| FUNCTION      | The @function definition used to define the key, if you defined one; for example, CREATE INDEX EMP1X ON EMP (@UPPER (NAME)) displays the @UPPER(NAME) in this column.                                                                                                                                                                                                  |
| DISTINCTCOUNT | <p>The number of distinct keys for the portion of the key from the first column up to the COLNO value.</p> <p>SYSKEYS has one entry for each prefix key of a multi-column index key. The first entry contains the DISTINCTCOUNT value for the first column in the key. The second entry contains DISTINCTCOUNT values for first two columns in the key, and so on.</p> |

## SYSADM.SYSOBAUTH

This table contains one row for each user granted execute privilege on an external function.

| Column Name | Description                                                         |
|-------------|---------------------------------------------------------------------|
| CREATOR     | The authorization ID of the user who created the external function. |

| Column Name | Description                                                                    |
|-------------|--------------------------------------------------------------------------------|
| NAME        | The name of the external function.                                             |
| GRANTEE     | The name of the user who has execute privilege on the external function.       |
| OBJAUTH     | Contains 'E' for execute to indicate the type of authorization for the object. |
| OBJTYPE     | Contains 'F' for external function to indicate the object type.                |

## SYSADM.SYSOBSYN

This table contains one row for each synonym created for an external function, stored command, or stored procedure.

| Column Name | Description                                                                                                             |
|-------------|-------------------------------------------------------------------------------------------------------------------------|
| NAME        | The synonym name.                                                                                                       |
| CREATOR     | The authorization ID of the user who created the synonym.                                                               |
| OBJNAME     | The name of the object.                                                                                                 |
| OBJCREATOR  | The authorization ID of the user who created the object.                                                                |
| OBJTYPE     | Contains 'F' for external function to indicate the object type. Contains 'P' for stored procedures and stored commands. |

## SYSADM.SYSPARTTRANS

This table contains information about in-doubt distributed transactions. Once a transaction is resolved, the row is removed from the table.

This table does not contain information about all in-doubt transactions while the database is active; it only contains rows about transactions that are in-doubt after a recovery.

| Column Name | Description                                                                                                                                                                                                                                                                                                                                                                               |
|-------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| ID1         | The global transaction ID.                                                                                                                                                                                                                                                                                                                                                                |
| ID2         | Reserved for future use.                                                                                                                                                                                                                                                                                                                                                                  |
| STATE       | The current status of the transaction. It can be any of the following: <ul style="list-style-type: none"> <li>• IDLE</li> <li>• PREPARED</li> <li>• COMMITTING</li> <li>• ABORTING</li> </ul>                                                                                                                                                                                             |
| PROTOCOL    | The protocol being used for the two-phase commit. Currently, SQLBase only uses the STANDARD protocol.                                                                                                                                                                                                                                                                                     |
| LASTMODTIME | The timestamp indicating when the transaction status was last modified.                                                                                                                                                                                                                                                                                                                   |
| DB          | The name of the commit server.                                                                                                                                                                                                                                                                                                                                                            |
| USERNAME    | The user name that is connected to the commit server.                                                                                                                                                                                                                                                                                                                                     |
| PASSWORD    | The (encrypted) password for connecting to the commit server.                                                                                                                                                                                                                                                                                                                             |
| SOURCE      | A string indicating how transaction information was entered into this table. Valid values are: <ul style="list-style-type: none"> <li>• NORMAL<br/>Indicates row was inserted while the database was active.</li> <li>• CRASH<br/>Indicates row was inserted just after crash recovery.</li> <li>• ROLLFORWARD<br/>Indicates row was inserted just after rollforward recovery.</li> </ul> |

## SYSADM.SYSPKCONSTRAINTS

This table contains the primary key columns of a table.

| Column Name | Description                                              |
|-------------|----------------------------------------------------------|
| CREATOR     | The authorization ID of the user who created the table.  |
| NAME        | The name of the table to which the primary key applies.  |
| PKCOLSEQNUM | The relative column number of the primary key column.    |
| COLNAME     | The name of the column to which the primary key applies. |

## SYSADM.SYSROWIDLISTS

This table lists information about result sets, and contains one row for each row ID. If SET RESTRICTION is set to ON, this table lists information only about the user's result sets.

| Column Name | Description                             |
|-------------|-----------------------------------------|
| NAME        | Name of row id list (result set) saved. |
| CREATOR     | Name of creator of above result set.    |

## SYSADM.SYSSYNONYMS

This table contains one row for each synonym of a table or view.

| Column Name | Description                                               |
|-------------|-----------------------------------------------------------|
| NAME        | Synonym for the table or view.                            |
| CREATOR     | The authorization ID of the creator of the synonym.       |
| TBNAME      | The name of the table or view.                            |
| TBCREATOR   | The authorization ID of the creator of the table or view. |

## SYSADM.SYSTABAUTH

This table contains the privileges of users for tables or views.

| Column Name | Description                                                                                                                                                                                                                                                                                                    |
|-------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| GRANTEE     | The authorization ID of the user who holds the privileges described in this row.                                                                                                                                                                                                                               |
| TCREATOR    | The authorization ID of the user who created the table or view on which privileges are held.                                                                                                                                                                                                                   |
| TTNAME      | The name of the table or view on which privileges are held. 'T' stands for target.                                                                                                                                                                                                                             |
| UPDATECOLS  | Contains a '*' if the user has update privileges on only some of the columns in the target table. The column names for which privileges are held are contained in the SYSCOLAUTH table. If the user holds no update privileges or if update is allowed on the entire table, then this column contains a blank. |
| ALTERAUTH   | Contains a 'Y' if the user is allowed to alter the target table. Otherwise it contains a blank.                                                                                                                                                                                                                |
| DELETEAUTH  | Contains a 'Y' if the user is allowed to delete rows from the target table or view. Otherwise it contains a blank.                                                                                                                                                                                             |
| INDEXAUTH   | Contains a 'Y' if the user is allowed to create or drop indexes on the target table. Otherwise it contains a blank.                                                                                                                                                                                            |
| INSERTAUTH  | Contains a 'Y' if the user is allowed to insert rows into the target table or view. Otherwise it contains a blank.                                                                                                                                                                                             |
| SELECTAUTH  | Contains a 'Y' if the user is allowed to read rows from the target table or view. Otherwise it contains a blank.                                                                                                                                                                                               |
| UPDATEAUTH  | Contains a 'Y' if the user is allowed to update rows in the target table or view. Otherwise it contains a blank.                                                                                                                                                                                               |

## SYSADM.SYSTABCONSTRAINTS

This table contains the table constraints.

| Column Name | Description                                             |
|-------------|---------------------------------------------------------|
| CREATOR     | The authorization ID of the user who created the table. |
| NAME        | The name of the table to which the constraint applies.  |

| Column Name  | Description                                                                                                                                                                                                         |
|--------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CONSTRAINT   | The name of the constraint which is being used by the database. This column also shows the name of the foreign key. For the primary key, it is set to 'PRIMARY'.                                                    |
| TYPE         | The type of constraint. This column is set to 'P' for a primary key, and 'F' for a foreign key constraint.                                                                                                          |
| DELETERULE   | The DELETE rule being followed by the database. This column is not used for primary key constraints.<br><br>This column can be set to:<br><br>CASCADE C<br>SET NULL N<br>RESTRICT R<br><br>The default is RESTRICT. |
| USRERRINSDEP | The user-specified error message number for an INSERT_DEPENDENT violation. If there was no user-specified error, this column is set to 0.                                                                           |
| USRERRUPDDEP | The user-specified error message number for an UPDATE_DEPENDENT violation. If there was no user-specified error, this column is set to 0.                                                                           |
| USRERRDELPAR | The user-specified error message number for a DELETE_PARENT violation. If there was no user-specified error, this column is set to 0.                                                                               |
| USRERRUPDPAR | The user-specified error message number for an UPDATE_PARENT violation. If there was no user-specified error, this column is set to 0.                                                                              |

SYSADM.SYSTABLES

This table contains one row for each table or view.

| Column Name | Description                                                                                 |
|-------------|---------------------------------------------------------------------------------------------|
| CREATOR     | The authorization ID of the user who created the table or view.                             |
| NAME        | The name of the table or view.                                                              |
| COLCOUNT    | The number of columns in the table or view.                                                 |
| REMARKS     | A user-specified comment about each row. The maximum length of a comment is 254 characters. |

| Column Name     | Description                                                                                                                                                                                                                                                                                                                                                                |
|-----------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| TYPE            | 'T' if the row describes a table or 'V' if the row describes a view.                                                                                                                                                                                                                                                                                                       |
| SYSTEM          | 'Y' if the table is system-defined; 'N' if it is not. 'U' denotes "Utility" for some system-defined tables.                                                                                                                                                                                                                                                                |
| SNUM            | A unique serial number that serves as a table ID. This is used by the UNLOAD DATABASE command.                                                                                                                                                                                                                                                                             |
| LABEL           | A user-specified label about each table and row. The maximum length of a label is 30 characters.                                                                                                                                                                                                                                                                           |
| PERCENTFREE     | The amount of free space left in each table row page when it is initially filled. The default value is 10 percent.                                                                                                                                                                                                                                                         |
| ROWCOUNT        | The number of rows in the table. This is maintained dynamically in the table's control page, but is only recorded in this table when you execute UPDATE STATISTICS.                                                                                                                                                                                                        |
| PAGECOUNT       | The number of base row pages in the table. This column is updated each time you run UPDATE STATISTICS. If you do not set any values yourself with the SET clause of this command, this column is the sum of the ROWPAGECOUNT, EXTENTPAGECOUNT, and LONGPAGECOUNT values.<br>Initial default value is 2.                                                                    |
| ROWPAGECOUNT    | The number of base row pages in the table. This includes any extent pages that may also be allocated. This is recorded and updated only when you run UPDATE STATISTICS.<br>Initial default value is 2.                                                                                                                                                                     |
| LONGPAGECOUNT   | The number of pages in the table allocated to store LONG VARCHAR columns. This is recorded and updated only when you run UPDATE STATISTICS.                                                                                                                                                                                                                                |
| EXTENTPAGECOUNT | The number of extent pages.                                                                                                                                                                                                                                                                                                                                                |
| FREESLOTS       | The total number of free slots in all data pages.<br><br>When a new page is allocated to a table, there is one entry in the slot table, which is called a free slot. As new rows are inserted into the page, more entries are added to the slot-tables and the space for each additional slot is carved out of the free slot. When a row is deleted its slot becomes free. |

| Column Name   | Description                                                                                                                                                                                                                                                                                                                             |
|---------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| USEDSPACE     | The total number of used kilobytes in all data pages, including user data, page header, etc.                                                                                                                                                                                                                                            |
| FREESPACE     | The total number of free kilobytes in all data pages used by free slots (holes). A large ratio of freespace to the usedspace calls for either a database reorganization or a load/unload. One possible reason for this can be that the average base row length is too large. The ideal ratio of freespace to the used space is small.   |
| AVGROWLEN     | The average row length in the table. This can differ significantly from the defined row length because SQLBase stores all columns as variable data regardless of the data type used in the column definition. Note that this is the row length that is stored on the base table pages, and therefore excludes all LONG VARCHAR columns. |
| AVGROWLONGLEN | The average length of all LONGVARCHAR columns stored in the table. This is zero (0) if no LONGVARCHAR columns exist.                                                                                                                                                                                                                    |
| GROUPNUM      | The group number for the table.                                                                                                                                                                                                                                                                                                         |
| TABLESCAN     | Not in use.                                                                                                                                                                                                                                                                                                                             |

SYSADM.SYSTRGCOLS

This table contains one row for every column on which an update trigger exists.

| Column Name | Description                                                                            |
|-------------|----------------------------------------------------------------------------------------|
| CREATOR     | The authorization ID of the creator of the trigger.                                    |
| NAME        | The name of the trigger.                                                               |
| TBSNUM      | The serial number of the trigger table.                                                |
| ACTIONTIME  | Whether the trigger is defined to activate before or after the attempted modification. |
| COLNO       | The column's relative position in the trigger table (starting with 1).                 |

## SYSADM.SYSTRIGGERS

This table contains one row for every trigger.

| Column Name  | Description                                                                                                                                                                                                                                              |
|--------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CREATOR      | The authorization ID of the creator of the trigger.                                                                                                                                                                                                      |
| NAME         | The name of the trigger.                                                                                                                                                                                                                                 |
| TBSNUM       | The serial number of the trigger table.                                                                                                                                                                                                                  |
| ACTIONTIME   | Whether the trigger is defined to activate before or after the attempted modification.                                                                                                                                                                   |
| TRIGGEREVENT | The event (INSERT, UPDATE, or DELETE) on which the trigger is designed to activate.                                                                                                                                                                      |
| OLDVALUENAME | The alias for the table containing the old column values.                                                                                                                                                                                                |
| NEWVALUENAME | The alias for the table containing the new column values.                                                                                                                                                                                                |
| FREQUENCY    | Whether the trigger is defined to activate once for each row or for each statement.                                                                                                                                                                      |
| SYSTEM       | 'Y' if the trigger is system-defined; 'N' if it is not.                                                                                                                                                                                                  |
| SPSNUM       | An integer column that refers to the serial number (SNUM in SYSCOMMANDS) of the stored procedure that is executed by a trigger.                                                                                                                          |
| TEXT         | The text of the stored or inline procedure.                                                                                                                                                                                                              |
| NUMCOLUMNS   | <p>The number of columns named in the UPDATE OF clause. The default is zero (0) which means that the trigger is activated by a modification attempt on any column.</p> <p>This is zero (0) for triggers defined on INSERT and DELETE operations.</p>     |
| ORDERVALUE   | The order sequence number of a trigger. The sequence number determines the order in which triggers are fired within a table, by event (using the INSERT, UPDATE, and DELETE commands), time (BEFORE and AFTER), and frequency (using STATEMENT and ROW). |
| STATUS       | 'Y' if the trigger is enabled; 'N' if it is disabled.                                                                                                                                                                                                    |

## SYSADM.SYSUSERAUTH

This table contains the database authority level of each user.

| Column Name  | Description                                                                                           |
|--------------|-------------------------------------------------------------------------------------------------------|
| NAME         | The authorization ID of each valid user of the database. A user is valid if he has CONNECT authority. |
| RESOURCEAUTH | 'Y' if the user can create or drop tables or 'G' if the user is SYSADM. Otherwise blank.              |
| DBAAUTH      | 'Y' if the user has DBA authority, a 'G' if the user is SYSADM. Otherwise blank.                      |
| PASSWORD     | The (encrypted) password associated with the authorization ID.                                        |

## SYSADM.SYSVIEWS

This table contains one or more rows for each view.

| Column Name | Description                                                                                                                                                                     |
|-------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| NAME        | The name of the view.                                                                                                                                                           |
| CREATOR     | The authorization ID of the creator of the view.                                                                                                                                |
| SEQNO       | The sequence number of this row. The first text portion of the view is on row one and successive rows have increasing values of SEQNO.                                          |
| CHECKFLAG   | Whether the CHECK option was specified in the CREATE VIEW command; 'Y' means yes and 'N' means no.                                                                              |
| TEXT        | The text of the CREATE VIEW command. This column is 250 characters long. If the CREATE VIEW command is longer than 40 characters, then each row contains a portion of the text. |

## System table operations

The system catalog tables are all owned by SYSADM. Users with the proper privileges can perform the following operations on system tables:

- Select rows
- Create a view

- Create a synonym
- Create an index on a column
- Add user-defined columns
- Drop user-defined columns
- Update user-defined columns

You cannot DROP a system table or system-defined column, nor can you INSERT or DELETE rows into or from a system table.

The privilege to perform any of the above operations must be granted explicitly to a user by the SYSADM or by a user with DBA authority. Users who do not have these privileges cannot access the system tables directly.

## System catalog views

You can create views of the system tables for users. Use the keyword USER in the view definition's search condition. For example, a view called TABLES could contain the names of all the tables that can be accessed with a given authorization ID; a view called COLUMNS could contain the names of all the columns in those tables, and so forth. SYSADM would own these views and other users could select from them but could not modify them.

Although you can add columns to a system catalog table, SQLBase does not maintain them. For example, the UNLOAD command ignores user-defined columns; it does not unload them.

# Server-independent system catalog views

Your applications can use the following views on the system catalog tables to access catalog information in a server-independent fashion.

These views are a common subset of the system catalog tables supported by most SQL database vendors. Commands such as LOAD and UNLOAD use these views to access non-SQLBase databases (such as DB2).

| View Name              | Columns                                                                                                    |
|------------------------|------------------------------------------------------------------------------------------------------------|
| SYSSQL.SYSCOLAUTH      | GRANTEE<br>CREATOR<br>TNAME<br>COLNAME                                                                     |
| SYSSQL.SYSCOLUMNS      | TBCREATOR<br>NAME<br>TBNAME<br>COLNO<br>COLTYPE<br>LENGTH<br>NULLS<br>UPDATES<br>REMARKS<br>SCALE<br>LABEL |
| SYSSQL.SYSCOMMANDS     | CREATOR<br>NAME<br>TYPE<br>SYSTEM<br>VALID<br>AUTORECOMPILE<br>TEXT                                        |
| SYSSQL.SYSDEPENDENCIES | DETCREATOR<br>DETNAME<br>DETTYPE<br>DEPCREATOR<br>DEPNAME<br>DEPTYPE                                       |

| View Name             | Columns                                                                                                            |
|-----------------------|--------------------------------------------------------------------------------------------------------------------|
| SYSSQL.SYSEVENTS      | CREATOR<br>NAME<br>TYPE<br>TYPEID<br>EVENTID<br>BEGINTIME<br>INTERVAL<br>TEXT                                      |
| SYSSQL.SYSEXECUTEAUTH | CREATOR<br>NAME<br>GRANTEE<br>USECREATORPRIV<br>USEGRANTEEPRIV                                                     |
| SYSSQL.SYSINDEXES     | TBCREATOR<br>NAME<br>TBNAME<br>CREATOR<br>UNIQUERULE<br>COLCOUNT<br>IXTYPE<br>CLUSTERRULE<br>IXSIZE<br>PERCENTFREE |
| SYSSQL.SYSKEYS        | IXCREATOR<br>IXNAME<br>COLNAME<br>COLNO<br>COLSEQ<br>ORDERING<br>FUNCTION                                          |
| SYSSQL.SYSOBJAUTH     | CREATOR<br>NAME<br>GRANTEE<br>OBJAUTH<br>OBJTYPE                                                                   |
| SYSSQL.SYSOBJSYN      | NAME<br>CREATOR<br>OBJNAME<br>OBJCREATOR<br>OBJTYPE                                                                |

| View Name           | Columns                                                                                                                       |
|---------------------|-------------------------------------------------------------------------------------------------------------------------------|
| SYSSQL.SYSPARTTRANS | ID<br>STATE<br>PROTOCOL<br>LASTMODTIME<br>DB<br>USERNAME<br>PASSWORD                                                          |
| SYSSQL.SYSSYNONYMS  | NAME<br>CREATOR<br>TBNAME<br>TBCREATOR                                                                                        |
| SYSSQL.SYSTABAUTH   | GRANTEE<br>TCREATOR<br>TTNAME<br>UPDATECOLS<br>ALTERAUTH<br>DELETEAUTH<br>INDEXAUTH<br>INSERTAUTH<br>SELECTAUTH<br>UPDATEAUTH |
| SYSSQL.SYSTABLES    | CREATOR<br>NAME<br>COLCOUNT<br>TYPE<br>REMARKS<br>PERCENTFREE<br>LABEL                                                        |

## MAIN database system tables

In addition to the standard system catalog tables, the MAIN database contains system catalog tables specific to partitioned databases and distributed transactions. These tables are owned by SYSADM and maintained by SQLBase.

| Table Name | Description                                                                     |
|------------|---------------------------------------------------------------------------------|
| CSVPARTS   | Contains information about all the participants in a distributed transaction.   |
| CSVTRANS   | Used by the distributed transaction's commit server to handle two-phase commit. |
| DEFAULTS   | Lists default storage group.                                                    |

| Table Name | Description                                    |
|------------|------------------------------------------------|
| DATABASES  | Lists all partitioned databases.               |
| STOGROUPS  | Lists storage groups.                          |
| AREAS      | Lists database areas.                          |
| STOAREAS   | Lists database areas and their storage groups. |
| EXTENTS    | Lists extents that have been used.             |
| FREEEXTS   | Lists extents that are available.              |
| CSVTRANSV  | Internal use only.                             |
| CSVPARTSV  | Internal use only.                             |

## SYSADM.CSVPARTS

This table contains information about all the participants in a transaction.

| Column Name | Description                                                   |
|-------------|---------------------------------------------------------------|
| ID1         | The global transaction ID.                                    |
| ID2         | Reserved for future use.                                      |
| DB          | The name of the commit server.                                |
| USERNAME    | The user name that is connected to the commit server.         |
| PASSWORD    | The (encrypted) password for connecting to the commit server. |

## SYSADM.CSVTRANS

This table is used by the commit server to record general information about the two-phase commit operation. There is one row for every distributed transaction.

| Column Name | Description                |
|-------------|----------------------------|
| ID1         | The global transaction ID. |
| ID2         | Reserved for future use.   |

| Column Name | Description                                                                                                                                                                              |
|-------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| STATE       | The current status of the transaction. It can be any of the following: <ul style="list-style-type: none"><li>• IDLE</li><li>• PREPARED</li><li>• COMMITTING</li><li>• ABORTING</li></ul> |
| PROTOCOL    | The protocol being used for the two-phase commit. Currently, SQLBase only uses the STANDARD protocol.                                                                                    |
| NUMPARTS    | The number of participants.                                                                                                                                                              |
| COMMITTIME  | Indicates when the transaction was committed.                                                                                                                                            |
| LASTMODTIME | A timestamp indicating when the transaction status was last modified.                                                                                                                    |

## SYSADM.DEFAULTS

This table contains information about default storage groups.

| Column Name | Description                 |
|-------------|-----------------------------|
| STOGROUP    | Default storage group name. |

## SYSADM.DATABASES

This table contains information about partitioned databases.

| Column Name | Description                                  |
|-------------|----------------------------------------------|
| NAME        | Partitioned database name.                   |
| STOGROUP    | Storage group associated with the database.  |
| LOGSTOGROUP | Storage group associated with the log files. |

## SYSADM.AREAS

This table contains information about database areas.

| Column Name | Description                                                                  |
|-------------|------------------------------------------------------------------------------|
| NAME        | Area name.                                                                   |
| AREAID      | Internal use only.                                                           |
| PATHNAME    | Path of the area file or device.                                             |
| AREASIZE    | Size of the area in bytes. Note that for the view, the size is in megabytes. |

## SYSADM.STOGROUPS

This table contains information about storage groups.

| Column Name | Description         |
|-------------|---------------------|
| NAME        | Storage group name. |

## SYSADM.STOAREAS

This table contains information about storage groups in association with areas.

| Column Name | Description         |
|-------------|---------------------|
| STOGROUP    | Storage group name. |
| AREANAME    | Database area name. |

## SYSADM.EXTENTS

This table contains information about database extents that have been used.

| Column Name | Description                                                                         |
|-------------|-------------------------------------------------------------------------------------|
| AREAID      | Internal use only.                                                                  |
| OFFSET      | Offset within the area, in bytes. Note that for the view, the size is in megabytes. |
| EXTSIZE     | Size of the extent, in bytes. Note that for the view, the size is in megabytes.     |
| DBNAME      | Name of the database associated with the extent.                                    |

| Column Name | Description        |
|-------------|--------------------|
| FILENAME    | Internal use only. |
| FILEPOS     | Internal use only. |

## SYSADM.FREEEXTS

This table contains information about available database extents.

| Column Name | Description                                                                          |
|-------------|--------------------------------------------------------------------------------------|
| AREAID      | Internal use only.                                                                   |
| OFFSET      | Offset within the area, in bytes. Note that for the view, the size is in megabytes.  |
| EXTSIZE     | Size of the free extent, in bytes. Note that for the view, the size is in megabytes. |

## MAIN database system views

The MAIN database contains system catalog views specific to distributed transactions and partitioned databases. These views are also owned by SYSSQL and maintained by SQLBase.

| Table Name | Description                                                      |
|------------|------------------------------------------------------------------|
| CSVPARTS   | ID<br>DB<br>USERNAME                                             |
| CSVTRANS   | ID<br>STATE<br>PROTOCOL<br>NUMPARTS<br>COMMITTIME<br>LASTMODTIME |
| DEFAULTS   | STOGROUP                                                         |
| DATABASES  | NAME<br>STOGROUP<br>LOGSTOGROUP                                  |
| STOGROUPS  | NAME                                                             |

| Table Name | Description                         |
|------------|-------------------------------------|
| AREAS      | NAME<br>AREASIZE<br>PATHNAME        |
| STOAREAS   | STOGROUP<br>AREANAME                |
| EXTENTS    | NAME<br>DBNAME<br>EXTSIZE<br>OFFSET |
| FREEEXTS   | OFFSET<br>EXTSIZE                   |

## MAIN stored commands

These stored commands manipulate the information in the tables of the MAIN database.

### DATABASES table:

```
store selname select name from databases
 where name = :dbname;

store selddb select stogroup from databases
 where name = :dbname;

store sellog select logstogroup from databases
 where name = :dbname;
```

### EXTENTS table:

```
store selsize select sum(extsize) from extents
 where dbname = :dbname and filename =
 :filename;

store selexts select areaid, offset, extsize
 from extents
 where dbname = :dbname and filename =
 :filename
 order by filepos;

store selexta select areaid, offset, extsize
 from extents
 where dbname = :dbname and filename =
 :filename
 order by areaid, offset;
```

```
store delfile delete from extents where dbname =
 :dbname
 and filename = :filename;
store inext insert into extents values
 (:areaid, :offset, :extsize,
 :dbname, :filename, :filepos);
```

**AREAS table:**

```
store selarea select pathname, areasize from
 areas where areaid = :areaid;
```

**FREEEXTS table:**

```
store selgt select f.areaid, f.offset, f.extsize
 from freeexts f, areas a, stoareas s
 where f.areaid > :areaid and f.extsize >=
 :extsize
 and s.stogroup = :stogroup and s.areaname
 = a.name
 and a.areaid = f.areaid order by f.areaid,
 f.extsize;

store selleg select f.areaid, f.offset,
 f.extsize
 from freeexts f, areas a, stoareas s
 where f.areaid <= :areaid and f.extsize >=
 :extsize
 and s.stogroup = :stogroup and s.areaname
 = a.name
 and a.areaid = f.areaid order by f.areaid,
 f.extsize;

store selsto select f.areaid, f.offset,
 f.extsize
 from freeexts f, areas a, stoareas s
 where f.extsize >= :extsize and s.stogroup
 = :stogroup
 and s.areaname = a.name and a.areaid =
 f.areaid
 order by f.extsize;

store selsame select offset, extsize from
 freeexts
 where areaid = :areaid and extsize >=
 :extsize
 order by extsize;
```

```

store delfree delete from freeexts where areaid
 = :areaid and
 offset = :offset;

store updfree update freeexts set offset =
 :newoffset,
 extsize = :newextsize where areaid =
 :areaid and
 offset = :offset;

store selcomb select offset, extsize from
 freeexts
 where areaid = :areaid and (offset +
 extsize = :offset or
 offset = :endoffset) order by offset;

store insfree insert into freeexts values
 (:areaid, :offset, :extsize);

```

## Client stored commands

```

store selaid select areaid from areas where areaid
 = :1;

store insare insert into areas values (:1, :2, :3,
 :4);

store insfex insert into freeexts values(:1, :2,
 :3);

store selasf select areasize, pathname from areas
 where name = :1;

store seleps select areas.areaaid from areas,
 extents
 where extents.offset + extents.extsize > :1
 and areas.name = :2 and areas.areaaid =
 extents.areaaid;

store selasz select areasize from areas where name
 = :1;

store updasz update areas set areasize = :1 where
 name = :2;

store selsta select areaname from stoareas where
 areaname = :1;

store selexa select extents.areaaid from extents,
 areas
 where areas.areaaid = extents.areaaid and
 areas.name = :1;

```

```
store delfex delete from freeexts
 where areaid = (select areaid from areas
 where name = :1);
store delare delete from areas where name = :1;
store insstg insert into stogroups values (:1);
store inssta insert into stoareas values (:1, :2);
store delsta delete from stoareas where stogroup =
 :1 and
 areaname = :2;
store selsgd select stogroup from databases where
 stogroup = :1;
store seldfs select stogroup from defaults;
store seldfl select stogroup from defaults where
 stogroup = :1;
store delstg delete from stogroups where name = :1;
store delasa delete from stoareas where stogroup =
 :1;
store deldfl delete from defaults;
store insdfl insert into defaults values (:1);
store selsgp select name from stogroups where name
 = :1;
store insdbs insert into databases values (:1, :2,
 :3);
store upddbds update databases set stogroup = :1
 where name = :2;
store upddbsl update databases set logstogroup = :1
 where name = :2;
store deldbs delete from databases where name = :1;
store selddb select name from databases where name
 = :1;
store selexf select filename from extents where
 dbname = :1;
store seladb select name from databases;
store selare select areaid from areas where name =
 :1;
store updfxs update freeexts set extsize = extsize
 - :1
 where areaid = :2 and offset + extsize > :3;
```

```
store delfxs delete from freeexts where extsize =
0;
```



## Appendix B

# SQLBase Procedures

---

This appendix lists the SQLBase-supplied procedures. SQLBase supplies the RECOMPILE and SYSADM.SYSPROC\_ALTTABTRIG procedures.

## SQLBase-supplied procedures

The SQLBase-provided procedures are not part of the *start.dbs* template database. Instead, the procedures are installed under a specific directory. Read the steps below for details.

### Loading SQLBase-supplied procedures

---

**Note:** To load SQLBase-supplied procedures, you must have SYSADM authority.

---

1. Start your server.
2. Start SQLTalk and connect to the database where you want to load the stored procedure(s).
3. Go to the following directory to locate the stored procedure(s) you want to load:
  - *recomp.sql* is found in the *sysproc* directory (This directory is installed under the directory where you install the client software.)
  - *rep\_trig.sql* is found in the Gupta directory
4. Run the *recomp.sql* and *rep\_trig.sql* scripts in the SQLTalk window or at the SQL> command line prompt. These scripts “load” the procedures to your specified database. The procedures are installed during SQLBase installation.

You must run *recomp.sql* or *rep\_trig.sql* against each database that requires the use of the stored procedure. For example, run *rep\_trig.sql* against all databases that contain the triggers you want to enable or disable. Note that you only need to run the script once for each database. You must be in the Gupta home directory (the default) when you run any procedure load scripts.

## RECOMPILE procedure

Use the RECOMPILE stored procedure to manually recompile a stored command. For information on stored procedures, see the *Procedures and Triggers* chapter in the *SQLBase SQL Language Reference*.

You should run the RECOMPILE procedure when a stored command is invalidated. A stored command becomes invalid when you alter its associated tables, such as when you drop an index or column. Even if the table change does not directly affect the stored command, SQLBase still invalidates it. You cannot use the stored command until you replace or recompile it.

The RECOMPILE procedure requires the following parameters:

- User id

This is the user name associated with the stored command.

- Stored command name

Enter the command name in capital letters. You can use the underscore ( \_ ) and percent ( % ) wildcard characters for pattern matching.

- Valid/Invalid flag

Enter **N** to designate invalid commands, or **Y** to designate valid commands. For more information on this flag, read the information under the SYSADM.SYSCOMMANDS table in *Appendix A, System Catalog Tables*.

- Input/Output values

These are a stored procedure's receive parameters as described in the *Procedures, Triggers, and Events* chapter in the *SQLBase SQL Language Reference*. There are three parameters, and each requires a value. You can enter a dummy value (for example, 0) to satisfy the execution requirement.

Unlike ALTER COMMAND, the EXECUTE RECOMPILE command actually recompiles a stored command. ALTER COMMAND sets the AUTORECOMPILE flag to ON; the stored command is not recompiled until it is executed and then only if it is invalid.

The RECOMPILE procedure does not alter the stored command's AUTORECOMPILE attribute (ON/OFF). You do not need to recompile system commands; SQLBase automatically recompiles them itself.

## SYSADM.SYSPROC\_ALTTABTRIG procedure

Use the SYSADM.SYSPROC\_ALTTABTRIG procedure to enable or disable all triggers defined on a table. This procedure requires the following parameters:

- User id

This is the user name associated with the table.

- Table name

Enter the table name in capital letters. You can use the underscore ( \_ ) and percent ( % ) wildcard characters for pattern matching.

- Trigger status

To change the status of the existing triggers defined on the table, Enter either ENABLE or DISABLE.

- Input/Output values

These are a stored procedure's receive parameters as described in the *Procedures, Triggers, and Events* chapter in the *SQLBase SQL Language Reference*. The procedure returns the number of triggers enabled/disabled by

this execution of the procedure. You can enter a dummy value (for example, 0) to satisfy the execution requirement.

## Appendix C

# System Utility Tables

---

This Appendix contains descriptions of the SQLBase system utility tables.

## System utilities table summary

The system utility tables are used for certain internal operations of SQLBase. A system utility table does not contain any database object definitions and the rows in the table are not created by any explicit DDL statements. The tables are owned by SYSADM and maintained by SQLBase.

| Table Name     | Description                              |
|----------------|------------------------------------------|
| SYSCOMMITORDER | Used for logging transaction commit logs |

### SYSCOMMITORDER

This table is used for logging transaction commit logs. If you call the *sqlset* function and specify the SQLPCLG parameter, you can turn commit logging on. For details, read the description of the *sqlset* function in the *SQL Application Programming Interface Reference*.

When commit logging is on, every database transaction in which data was modified is logged in the SYSCOMMIT ORDER table.

---

**Note:** This table is primarily for internal use and should not be modified. You cannot drop this table or delete columns from it. If you are using SQLBase for Gupta replication, manipulating this table or the SQLPCLG parameter may result in erroneous replication behavior.

---

| Column Name    | Description                                         |
|----------------|-----------------------------------------------------|
| CommitSequence | The sequence number for the committing transaction. |
| TransactionID  | The transaction ID of the committing transaction.   |

# Glossary

---

***access path***—The path used to get the data specified in a SQL command. An access path can involve an index or a sequential search (table scan), or a combination of the two. Alternate paths are judged based on the efficiency of locating the data.

***aggregate function***—A SQL operation that produces a summary value from a set of values.

***alias***—An alternative name used to identify a database object.

***API (application programming interface)***—A set of functions that a program uses to access a database.

***application***—A program written by or for a user that applies to the user's work. A program or set of programs that perform a task. For example, a payroll system.

***argument***—A value entered in a command that defines the data to operate on or that controls execution. Also called parameter or operand.

***arithmetic expression***—An expression that contains operations and arguments that can be reduced to a single numeric value.

***arithmetic operator***—A symbol used to represent an arithmetic operation, such as the plus sign (+) or the minus sign (-).

***attribute***—A characteristic or property. For example, the data type or length of a row. Sometimes, attribute is used as a synonym for column or field.

***audit file***—A log file that records output from an audit operation.

***audit message***—A message string that you can include in an audit file

***audit operation***—A SQLBase operation that logs database activities and performance, writing output to an audit file. For example, you can monitor who logs on to a database and what tables they access, or record command execution time.

***authorization***—The right granted to a user to access a database.

***authorization-ID***—A unique name that identifies a user. Associated to each authorization-id is a password. Abbreviated auth-id. Also called username.

***back-end***—See database server.

***backup***—To copy information onto a diskette, fixed disk, or tape for record keeping or recovery purposes.

***base table***—The permanent table on which a view is based. A base table is created with the CREATE TABLE command and does not depend on any other table. A base table has its description and its data physically stored in the database. Also called underlying table.

***bindery***—A NetWare database that contains information about network resources such as a SQLBase database server.

***bind variable***—A variable used to associate data to a SQL command. Bind variables can be used in the VALUES clause of an INSERT command, in a WHERE clause, or in the SET clause of an UPDATE command. Bind variables are the mechanism to transmit data between an application work area and SQLBase. Also called into variable or substitution variable.

***browse***—A mode where a user queries some of a database without necessarily making additions or changes. In a browsing application, a user needs to examine data before deciding what to do with it. A browsing application allows the user to scroll forward and backward through data.

***buffer***—A memory area used to hold data during input/output operations.

***C/API***—A language interface that lets a programmer develop a database application in the C programming language. The C/API has functions that a programmer calls to access a database using SQL commands.

***cache***—A temporary storage area in computer memory for database pages being accessed and changed by database users. A cache is used because it is faster to read and write to computer memory than to a disk file.

***Cartesian product***—In a join, all the possible combinations of the rows from each of the tables. The number of rows in the Cartesian product is equal to the number of rows in the first table times the number of rows in the second table, and so on. A Cartesian product is the first step in joining tables. Once the Cartesian product has been formed, the rows that do not satisfy the join conditions are eliminated.

---

***cascade***—A delete rule which specifies that changing a value in the parent table automatically affects any related rows in the dependent table.

***case sensitive***—A condition in which names must be entered in a specific lower-case, upper-case, or mixed-case format to be valid.

***cast***—The conversion between different data types that represent the same data.

***CHAR***—A column data type that stores character strings with a user-specified length. SQLBase stores CHAR columns as variable-length strings. Also called VARCHAR.

***character***—A letter, digit, or special character (such as a punctuation mark) that is used to represent data.

***character string***—A sequence of characters treated as a unit.

***checkpoint***—A point at which database changes older than the last checkpoint are flushed to disk. Checkpoints are needed to ensure crash recovery.

***clause***—A distinct part of a SQL command, such as the WHERE clause; usually followed by an argument.

***client***—A computer that accesses shared resources on other computers running as servers on the network. Also called front-end or requester.

***column***—A data value that describes one characteristic of an entity. The smallest unit of data that can be referred to in a row. A column contains one unit of data in a row of a table. A column has a name and a data type. Sometimes called field or attribute.

***command***—A user request to perform a task or operation. In SQLTalk, each command starts with a name, and has clauses and arguments that tailor the action that is performed. A command can include limits or specific terms for its execution, such as a query for names and addresses in a single zip code. Sometimes called statement.

***commit***—A process that causes data changed by an application to become part of the physical database. Locks are freed after a commit (except when cursor-context preservation is on). Before changes are stored, both the old and new data exist so that changes can be stored or the data can be restored to its prior state.

***commit server***—A database server participating in a distributed transaction, that has commit service enabled. It logs information about the distributed transaction and assists in recover after a network failure.

***composite primary key***—A primary key made up of more than one column in a table.

***concatenated key***—An index that is created on more than one column of a table. Can be used to guarantee that those columns are unique for every row in the table and to speed access to rows via those columns.

***concatenation***—Combining two or more character strings into a single string.

***concurrency***—The shared use of a database by multiple users or application programs at the same time. Multiple users can execute database transactions simultaneously without interfering with each other. The database software ensures that all users see correct data and that all changes are made in the proper order.

***configure***—To define the features and settings for a database server or its client applications.

***connect***—To provide a valid authorization-id and password to log on to a database.

***connection handle***—Used to create multiple, independent connections. An application must request a connection handle before it opens a cursor. Each connection handle represents a single transaction and can have multiple cursors. An application may request multiple connection handles if it is involved in a sequence of transactions.

***consistency***—A state that guarantees that all data encountered by a transaction does not change for the duration of a command. Consistency ensures that uncommitted updates are not seen by other users.

***constant***—Specifies an unchanging value. Also called literal.

***control file***—An ASCII file containing information to manage segmented load/unload files.

***cooperative processing***—Processing that is distributed between a client and a server in a such a way that each computer works on the parts of the application that it is best at handling.

***coordinator***—The application that initiates a distributed transaction.

***correlated subquery***—A subquery that is executed once for each row selected by the outer query. A subquery cannot be evaluated independently because it depends on the outer query for its results. Also called a repeating query. Also see subquery and outer query.

***correlation name***—A temporary name assigned to a table in an UPDATE, DELETE, or SELECT command. The correlation name and column name are combined to refer to a column from a specific table later in the same command. A correlation name is used when a reference to a column name could be ambiguous. Also called range variable.

---

**crash recovery**—The procedures that SQLBase uses automatically to bring a database to a consistent state after a failure.

**CRC (Cyclic Redundancy Check)**—A technique that makes unauthorized changes to a database page detectable. SQLBase appends an extra bit sequence to every database page called a Frame Check Sequence (FCS) that holds redundant information about the page. When SQLBase reads a database page, it checks the FCS to detect unauthorized changes.

**current row**—The latest row of the active result set which has been fetched by a cursor. Each subsequent fetch retrieves the next row of the active result set.

**cursor**—The term cursor refers to one of the following definitions:

- The position of a row within a result table. A cursor is used to retrieve rows from the result table. A named cursor can be used in the CURRENT OF clause or the ADJUSTING clause to make updates or deletions.
- A work space in memory that is used for gaining access to the database and processing a SQL command. This work space contains the return code, number of rows, error position, number of select list items, number of bind variables, rollback flag, and the command type of the current command.
- When the cursor belongs to an explicit connection handle that is created using the SQL/API function call *sqlcch* or the SQLTalk BEGIN CONNECTION command, it identifies a task or activity within a transaction. The task or activity can be compiled/executed independently within a single connection thread.

Cursors can be associated with specific connection handles, allowing multiple transactions to the same database within a single application. When this is implemented, only one user is allowed per transaction.

- When a cursor belongs to an implicit connection handle created using the SQL/API function call *sqlcnc* or *sqlcnr*, or the SQLTalk CONNECT command, the cursor applies to an application in which you are connecting the cursor to a specific database that belongs to a single transaction.

**cursor-context preservation**—A feature of SQLBase where result sets are maintained after a COMMIT. A COMMIT does not destroy an active result set (cursor context). This enables an application to maintain its position after a COMMIT, INSERT, or UPDATE. For fetch operations, locks are kept on pages required to maintain the fetch position.

**cursor handle**—Identifies a task or activity within a transaction. When a connection handle is included in a function call to open a new cursor, the function call returns a cursor handle. The cursor handle can be used in subsequent SQL/API calls to identify the connection thread. A cursor handle is always part of a specific transaction and cannot be used in multiple transactions. However, a

cursor handle can be associated with a specific connection handle. The ability to have multiple transactions to the same database within a single application is possible by associating cursor handles with connection handles.

***Cursor Stability (CS)***—The isolation level where a page acquires a shared lock on it only while it is being read (while the cursor is on it). A shared lock is dropped as the cursor leaves the page, but an exclusive lock (the type of lock used for an update) is retained until the transaction completes. This isolation level provides higher concurrency than Read Repeatability, but consistency is lower.

***data dictionary***—See system catalog.

***data type***—Any of the standard forms of data that SQLBase can store and manipulate. An attribute that specifies the representation for a column in a table. Examples of data types in SQLBase are CHAR (or VARCHAR), LONG VARCHAR (or LONG), NUMBER, DECIMAL (or DEC), INTEGER (or INT), SMALLINT, DOUBLE PRECISION, FLOAT, REAL, DATETIME (or TIMESTAMP), DATE, TIME.

***database***—A collection of interrelated or independent pieces of information stored together without unnecessary redundancy. A database can be accessed and operated upon by client applications such as SQLTalk.

***database administrator (DBA)***—A person responsible for the design, planning, installation, configuration, control, management, maintenance, and operation of a DBMS and its supporting network. A DBA ensures successful use of the DBMS by users.

A DBA is authorized to grant and revoke other users' access to a database, modify database options that affect all users, and perform other administrative functions.

***database area***—A database area corresponds to a file. These areas can be spread across multiple disk volumes to take advantage of parallel disk input/output operations.

***database management system (DBMS)***—A software system that manages the creation, organization, and modification of a database and access to data stored within it. A DBMS provides centralized control, data independence, and complex physical structures for efficient access, integrity, recovery, concurrency, and security.

***database object***—A table, view, index, synonym or other object created and manipulated through SQL.

***database server***—A DBMS that a user interacts with through a client application on the same or a different computer. Also called back-end.

***DATE***—A column data type in SQL that represents a date value as a three-part value (day, month, and year).

---

***date/time value***—A value of the data type DATE, TIME, or TIMESTAMP.

***DCL (Data Control Language)***—SQL commands that assign database access privileges and security such as GRANT and REVOKE.

***DDL (Data Definition Language)***—SQL commands that create and define database objects such as CREATE TABLE, ALTER TABLE, and DROP TABLE.

***deadlock***—A situation when two transactions, each having a lock on a database page, attempt to acquire a lock on the other's database page. One type of deadlock is where each transaction holds a shared lock on a page and each wishes to acquire an exclusive lock. Also called deadly embrace.

***DECIMAL***—A column data type that contains numeric data with a decimal point. Also called DEC.

***default***—An attribute, value, or setting that is assumed when none is explicitly specified.

***delimited identifier***—An identifier enclosed between two double quote characters (") because it contains reserved words, spaces, or special characters.

***delimiter***—A character that groups or separates items in a command.

***dependent object***—An object whose existence depends on another object.

For example, if a stored procedure calls an external function, the stored procedure is the dependent object of the external function, since its existence depends on the external function.

***dependent table***—The table containing the foreign key.

***determinant object***—An object that determines the existence of another object.

For example, if a stored procedure calls an external function, the external function is the determinant object, since it determines the existence of the stored procedure.

***digital signature***—A unique binary number generated by an algorithm that identifies the content of a larger block of bytes.

***dirty page***—A database page in cache that has been changed but has not been written back to disk.

***distributed database***—A database whose objects reside on more than one system in a network of systems and whose objects can be accessed from any system in the network.

***distributed transaction***—Coordinates SQL statements among multiple databases that are connected by a network.

***DLL (Dynamic Link Library)***—A program library written in C or assembler that contains related modules of compiled code. The functions in a DLL are not read until run-time (dynamic linking).

***DML (Data Manipulation Language)***—SQL commands that change data such as INSERT, DELETE, UPDATE, COMMIT, and ROLLBACK.

***DOUBLE PRECISION***—A column data type that stores a floating point number.

***DQL (Data Query Language)***—The SQL SELECT command, which lets a user request information from a database.

***duplicates***—An option used when creating an index for a table that specifies whether duplicate values are allowed for a key.

***embedded SQL***—SQL commands that are embedded within a program, and are prepared during precompilation and compilation before the program is executed. After a SQL command is prepared, the command itself does not change (although values of host variables specified within the command can change). Also called static SQL.

***encryption***—The transformation of data into a form unreadable by anyone without a decryption key or password. Encryption ensures privacy by keeping information hidden from anyone for whom it is not intended, even those who can see the encrypted data. Unencrypted data is called plain text; encrypted data is called cipher text.

***engine***—See database server.

***entity***—A person, place, or thing represented by a table. In a table, each row represents an entity.

***equijoin***—A join where columns are compared on the basis of equality, and all the columns in the tables being joined are included in the results.

***Ethernet***—A LAN with a bus topology (a single cable not connected at the ends). When a computer wants to transmit, it first checks to see if another computer is transmitting. After a computer transmits, it can detect if a collision has happened. Ethernet is a broadcast network and all computers on the network hear all transmissions. A computer selects only those transmissions addressed to it.

***exclusive lock (X-lock)***—An exclusive lock allows only one user to have a lock on a page at a time. An exclusive lock prevents another user from acquiring a lock until the exclusive lock is released. Exclusive locks are placed when a page is to be modified (such as for an UPDATE, INSERT, or DELETE).

An exclusive lock differs from a shared lock because it does not permit another user to place any type of lock on the same data.

---

***expression***—An item or a combination of items and operators that yield a single value. Examples are column names which yield the value of the column in successive rows, arithmetic expressions built with operators such as + or - that yield the result of performing the operation, and functions which yield the value of the function for its argument.

***extent page***—A database page used when a row is INSERTed that is longer than a page or when a row is UPDATed and there is not enough space in the original page to hold the data.

***external function***—A user-defined function that resides in an “external” DLL (Dynamic Link Library) invoked within a SQLBase stored procedure.

***field***—See column.

***file server***—A computer that allows network users to store and share information.

***FLOAT***—A column data type that stores floating point numbers.

***floating point***—A number represented as a number followed by an exponent designator (such as 1.234E2, -5.678E2, or 1.234E-2). Also called E-notation or scientific notation.

***foreign key***—Foreign keys logically connect different tables. A foreign key is a column or combination of columns in one table whose values match a primary key in another table. A foreign key can also be used to match a primary key within the same table.

***front-end***—See client.

***function***—A predefined operation that returns a single value per row in the output result table.

***grant***—That act of a system administrator to permit a user to make specified use of a database. A user may be granted access to an entire database or specific portions, and have unlimited or strictly-limited power to display, change, add, or delete data.

***GUI (Graphical User Interface)***—A graphics-based user interface with windows, icons, pull-down menus, a pointer, and a mouse. Microsoft Windows is an example of a graphical user interface.

***history file***—Contains previous versions of changed database pages. Used when read-only (RO) isolation level is enabled.

***host language***—A program written in a language that contains SQL commands.

***identifier***—The name of a database object.

***index***—A data structure associated with a table used to locate a row without scanning an entire table. An index has an entry for each value found in a table's indexed column or columns, and pointers to rows having that value. An index is logically ordered by the values of a key. Indexes can also enforce uniqueness on the rows in a table.

***INTEGER***—A column data type that stores a number without a decimal point. Also call INT.

***isolation level***—The extent to which operations performed by one user can be affected by (are isolated from) operations performed by another user. The isolation levels are Read Repeatability (RR), Cursor Stability (CS), Release Locks (RL), and Read Only (RO).

***JDBC (Java Database Connectivity)***—An industry standard for database-independent connectivity between Java and a range of databases. The JDBC provides a call-level API for SQL-based database access.

***join***—A query that retrieves data from two or more tables. Rows are selected when columns from one table match columns from another table. See also Cartesian product, self-join, equijoin, natural join, theta join, and outer join.

***key***—A column or a set of columns in an index used to identify a row. A key value can be used to locate a row.

***keyword***—One of the predefined words in a command language.

***local area network (LAN)***—A collection of connected computers that share data and resources, and access other networks or remote hosts. Usually, a LAN is geographically confined and microcomputer-based.

***lock***—To temporarily restrict other users access to data to maintain consistency. Locking prevents data from being modified by more than one user at a time and prevents data from being read while being updated. A lock serializes access to data and prevents simultaneous updates that might result in inconsistent data. See shared lock (S-lock) and exclusive lock (X-lock).

***logical operator***—A symbol for a logical operation that connects expressions in a WHERE or HAVING clause. Examples are AND, OR, and NOT. An expression formed with logical operators evaluates to either TRUE or FALSE. Logical operators define or limit the information sought. Also called Boolean operator.

***LONG VARCHAR***—In SQL, a column data type where the value can be longer than 254 bytes. The user does not specify a length. SQLBase stores LONG VARCHAR columns as variable-length strings. Also called LONG.

***mathematical function***—An operation such as finding the average, minimum, or maximum value of a set of values.

---

**media recovery**—Restoring data from backup after events such as a disk head crash, operating system crash, or a user accidentally dropping a database object.

**message buffer**—The input message buffer is allocated on both the client computer and the database server. The database server builds an input message in this buffer on the database server and sends it across the network to a buffer on the client. It is called an input message buffer because it is input from the client's point of view.

The output message buffer is allocated on both the client computer and on the database server. The client builds an output message in this buffer and sends it to a buffer on the database server. It is called an output message buffer because it is output from the client's point of view.

**modulo**—An arithmetic operator that returns an integer remainder after a division operation on two integers.

**multi-user**—The ability of a computer system to provide its services to more than one user at a time.

**natural join**—An equijoin where the value of the columns being joined are compared on the basis of equality. All the columns in the tables are included in the results but only one of each pair of joined columns is included.

**NDS (NetWare Directory Services)**—A network-wide directory included with NetWare 4.x, that provides global access to all network resources, regardless of their physical location. The directory is accessible from multiple points by network users, services and applications.

**nested query**—See subquery.

**NetWare**—The networking components sold by Novell. NetWare is a collection of data link drivers, a transport protocol stack, client computer software, and the NetWare server operating system. NetWare runs on Token Ring, Ethernet, and ARCNET.

**NetWare 386**—A server operating system from Novell for computers that controls system resources on a network.

**NLM (NetWare Loadable Module)**—An NLM is a NetWare program that you can load into or unload from server memory while the server is running. When loaded, an NLM is part of the NetWare operating system. When unloaded, an NLM releases the memory and resources that were allocated for it.

**null**—A value that indicates the absence of data. Null is not considered equivalent to zero or to blank. A value of null is not considered to be greater than, less than, or equivalent to any other value, including another value of null.

**NUMBER**—A column data type that contains a number, with or without a decimal point and a sign.

**numeric constant**—A fixed value that is a number.

**ODBC**—The Microsoft Open DataBase Connectivity (ODBC) standard, which is an application programming interface (API) specification written by Microsoft. It calls for all client applications to write to the ODBC standard API and for all database vendors to provide support for it. It then relies on third-party database drivers or access tools that conform to the ODBC specification to translate the ODBC standard API calls generated by the client application into the database vendor's proprietary API calls.

**operator**—A symbol or word that represents an operation to be performed on the values on either side of it. Examples of operators are: arithmetic (+, -, \*, /), relational (=, !=, >, <, >=, <=), and logical (AND, OR, NOT).

**optimization**—The determination of the most efficient access strategy for satisfying a database access.

**outer join**—A join in which both matching and non-matching rows are returned. Each preserved row is joined to an imaginary row in the other table in which all the fields are null.

**outer query**—When a query is nested within another query, the main query is called the outer query and the inner query is called the subquery. An outer query is executed once for each row selected by the subquery. A subquery cannot be evaluated independently but that depends on the outer query for its results. Also see subquery.

**page**—The physical unit of disk storage that SQLBase uses to allocate space to tables and indexes.

**parent table**—The table containing the primary key.

**parse**—To examine a command to make sure that it is properly formed and that all necessary information is supplied.

**partitioning**—A method of setting up separate user areas to maximize disk space. Databases can be stretched across several different network partitions.

**password**—A sequence of characters that must be entered to connect to a database. Associated to each password is an authorization-id.

**picture**—A string of characters used to format data for display.

**precedence**—The default order in which operations are performed in an expression.

**precision**—The maximum number of digits in a column.

---

***precompilation***—Processing of a program containing SQL commands or procedures that takes place before compilation. SQL commands are replaced with statements that are recognized by the host language compiler. Output from precompilation includes source code that can be submitted to the compiler.

***predicate***—An element in a search condition that expresses a comparison operation that states a set of criteria for the data to be returned by a query.

***primary key***—The columns or set of columns that are used to uniquely identify each row in a table. All values for a key are unique and non-null.

***privilege***—A capability given to a user to perform an action.

***procedure***—A named set of SAL or SQL statements that can contain flow control language. You compile a procedure for immediate and/or later execution.

***query***—A request for information from a database, optionally based on specific conditions. For example, a request to list all customers whose balance is greater than \$1000. Queries are issued with the SELECT command.

***Read Only (RO)***—The isolation level where pages are not locked, and no user has to wait. This gives the user a snapshot view of the database at the instant that the transaction began. Data cannot be updated while in the read-only isolation level.

***Read Repeatability (RR)***—The isolation level where if data is read again during a transaction, it is guaranteed that those rows would not have changed. Rows referenced by the program cannot be changed by other programs until the program reaches a commit point. Subsequent queries return a consistent set of results (as though changes to the data were suspended until all the queries finished). Other users will not be able to update any pages that have been read by the transaction. All shared locks and all exclusive locks are retained on a page until the transaction completes. Read repeatability provides maximum protection from other active application programs. This ensures a high level of consistency, but lowers concurrency. SQLBase default isolation level.

***REAL***—A column data type that stores a single-precision number.

***record***—See row.

***recovery***—Rebuilding a database after a system failure.

***referential cycle***—Tables which are dependents of one another.

***referential integrity***—Guarantees that all references from one database table to another are valid and accurate. Referential integrity prevents problems that occur because of changes in one table which are not reflected in another.

***relation***—See table.

**relational database**—A database that is organized and accessed according to relationships between data items. A relational database is perceived by users as a collection of tables.

**relational operator**—A symbol (such as =, >, or <) used to compare two values. Also called comparison operator.

**Release Locks (RL)**—With the Cursor Stability isolation level, when a reader moves off a database page, the shared lock is dropped. However, if a row from the page is still in the message buffer, the page is still locked.

In contrast, the Release Lock (RL) isolation level increases concurrency. By the time control returns to the application, all shared locks have been released.

**repeating query**—See correlated subquery.

**requester**—See client.

**restore**—Copying a backup of a database or its log files to a database directory.

**restriction mode**—In restriction mode, the result set of one query is the basis for the next query. Each query further restricts the result set. This continues for each subsequent query.

**result set mode**—Normally, result table rows are displayed and scrolled off the screen. In result set mode, the rows of the result table are available for subsequent scrolling and retrieval.

**result table**—The set of rows retrieved from one or more tables or views during a query. A cursor allows the rows to be retrieved one by one.

**revoke**—The act of withdrawing a user's permission to access a database.

**rollback**—To restore a database to the condition it was in at its last COMMIT. A ROLLBACK cancels a transaction and undoes any changes that it made to the database. All locks are freed unless cursor-context preservation is on.

**rollforward**—Reapplying changes to a database. The transaction log contains the entries used for rollforward.

**router**—A client application talks to a SQLBase server through a router program. The router enables a logical connection between a client and the server. Once this connection is established on the LAN, the client application uses the router program to send SQL requests to the server and to receive the results.

**row**—A set of related columns that describe a specific entity. For example, a row could contain a name, address, telephone number. Sometimes called record or tuple.

---

**ROWID**—A hidden column associated with each row in a SQLBase table that is an internal identifier for the row. The ROWID can be retrieved like any other column.

**ROWID validation**—A programming technique that ensures that a given row that was SELECTed has not been changed or deleted by another user during a session. When a row is updated, the ROWID is changed.

**SAP (Service Advertisement Protocol)**—A NetWare protocol that resources (such as database servers) use to publicize their services and addresses on a network.

**savepoint**—An intermediate point within a transaction to which a user can later ROLLBACK to cancel any subsequent commands, or COMMIT to complete the commands.

**scale**—The number of digits to the right of the decimal point in a number.

**search condition**—A criterion for selecting rows from a table. A search condition appears in a WHERE clause and contains one or more predicates.

**search**—To scan one or more columns in a row to find rows that have a certain property.

**self-join**—A join of a table with itself. The user assigns the two different correlation names to the table that are used to qualify the column names in the rest of the query.

**self-referencing table**—A table that has foreign and primary keys with matching values within the same table.

**server**—A computer on a network that provides services and facilities to client applications.

**SHA (Secure Hash Algorithm)**—A hash algorithm published by the United States government that SQLBase uses to detect unauthorized changes to a database page. SHA produces a condensed representation of a database page called a message digest that is used to generate a digital signature. When SQLBase reads a page encrypted with SHA, it verifies the signature. Any unauthorized changes to the page results in a different message digest and the signature will fail to verify. It is extremely unlikely to find a page that corresponds to a given message digest, or to find two different pages which produce the same message digest.

**shared lock (S-lock)**—A shared lock permits other users to read data, but not to change it. A shared lock lets users read data concurrently, but does not let a user acquire an exclusive lock on the data until all the users' shared locks have been released. A shared lock is placed on a page when the page is read (during a SELECT). At a given time, more than one user can have a shared lock placed on a page. The timing of the release of a shared lock depends on the isolation level.

A shared lock differs from an exclusive lock because it permits more than one user to place a lock on the same data.

***single-user***—A computer system that can only provide its services to one user at a time.

***SMALLINT***—A column data type that stores numbers without decimal points.

***socket***—An identifier that Novell's IPX (Internetwork Packet Exchange) uses to route packets to a specific program.

***SPX (Sequenced Packet Exchange)***—A Novell communication protocol that monitors network transmissions to ensure successful delivery. SPX runs on top of Novell's IPX (Internetwork Packet Exchange).

***SQL (Structured Query Language)***—A standard set of commands used to manage information stored in a database. These commands let users retrieve, add, update, or delete data. There are four types of SQL commands: Data Definition Language (DDL), Data Manipulation Language (DML), Data Query Language (DQL), and Data Control Language (DCL). SQL commands can be used interactively or they can be embedded within an application program. Pronounced ess-que-ell or sequel.

***SQLBase***—A relational DBMS that lets users access, create, and update data.

***SQLTalk***—SQLTalk is an interactive user interface for SQLBase that is used to manage a relational database. SQLTalk has a complete implementation of SQL and many extensions. SQLTalk is a client application.

***static SQL***—See embedded SQL.

***statistics***—Attributes about tables such as the number of rows or the number of pages. Statistics are used during optimization to determine the access path to a table.

***storage group***—A list of database areas. Storage groups provide a means to allow databases or tables to be stored on different volumes.

***stored procedure***—A precompiled procedure that is stored on the backend for future execution.

***string delimiter***—A symbol used to enclose a string constant. The symbol is the single quote (').

***string***—A sequence of characters treated as a unit of data.

***subquery***—A SELECT command nested within the WHERE or HAVING clause of another SQL command. A subquery can be used anywhere an expression is allowed if the subquery returns a single value. Sometimes called a nested query. Also called subselect. See also correlated subquery.

---

***synonym***—A name assigned to a table, view, external function that may be then used to refer to it. If you have access to another user's table, you may create a synonym for it and refer to it by the synonym alone without entering the user's name as a qualifier.

***syntax***—The rules governing the structure of a command.

***system catalog***—A set of tables SQLBase uses to store metadata. System catalog tables contain information about database objects, privileges, events, and users. Also called data dictionary.

***system keywords***—Keywords that can be used to retrieve system information in commands.

***table***—The basic data storage structure in a relational database. A table is a two-dimensional arrangement of columns and rows. Each row contains the same set of data items (columns). Sometimes called a relation.

***table scan***—A method of data retrieval where a DBMS directly searches all rows in a table sequentially instead of using an index.

***theta join***—A join that uses relational operators to specify the join condition.

***TIME***—A column data type in the form of a value that designates a time of day in hours, minutes, and possibly seconds (a two- or three-part value).

***timeout***—A time interval allotted for an operation to occur.

***TIMESTAMP***—A column data type with a seven-part value that designates a date and time. The seven parts are year, month, day, hour, minutes, seconds, and microseconds (optional). The format is:

yyyy-mm-dd-hh.mm.ss.nnnnnn

***token***—A character string in a specific format that has some defined significance in a SQL command.

***Token-Ring***—A LAN with ring topology (cable connected at the ends). A special data packet called a token is passed from one computer to another. When a computer gets the token, it can attach data to it and transmit. Each computer passes on the data until it arrives at its destination. The receiver marks the message as being received and sends the message on to the next computer. The message continues around the ring until the sender receives it and frees the token.

***tokenized error message***—An error message formatted with tokens in order to provide users with more informational error messages. A tokenized error message contains one or more variables that SQLBase substitutes with object names (tokens) when it returns the error message to the user.

**transaction**—A logically-related sequence of SQL commands that accomplishes a particular result for an application. SQLBase ensures the consistency of data by verifying that either all the data changes made during a transaction are performed, or that none of them are performed. A transaction begins when the application starts or when a COMMIT or ROLLBACK is executed. The transaction ends when the next COMMIT or ROLLBACK is executed. Also called logical unit of work.

**transaction log**—A collection of information describing the sequence of events that occur while running SQLBase. The information is used for recovery if there is a system failure. A log includes records of changes made to a database. A transaction log in SQLBase contains the data needed to perform rollbacks, crash recovery, and media recovery.

**trigger**—Activates a stored procedure that SQLBase automatically executes when a user attempts to change the data in a table, such as on a DELETE or UPDATE command.

**two-phase commit**—The protocol that coordinates a distributed transaction commit process on all participating databases.

**tuple**—See row.

**unique key**—One or more columns that must be unique for each row of the table. An index that ensures that no identical key values are stored in a table.

**username**—See authorization-id.

**value**—Data assigned to a column, a constant, a variable, or an argument.

**VARCHAR**—See CHAR.

**variable**—A data item that can assume any of a given set of values.

**view**—A logical representation of data from one or more base tables. A view can include some or all of the columns in the table or tables on which it is defined. A view represents a portion of data generated by a query. A view is derived from a base table or base tables but has no storage of its own. Data for a view can be updated in the same manner as for a base table. Sometimes called a virtual table.

**wildcard**—Characters used in the LIKE predicate that can stand for any one character (the underscore \_) or any number of characters (the percent sign %) in pattern-matching.

**Windows**—A graphical user interface from Microsoft.

With Windows, commands are organized in lists called menus. Icons (small pictures) on the screen represent applications. A user selects a menu item or an icon by pointing to it with a mouse and clicking.

---

Applications run in windows that can be resized and relocated. A user can run two or more applications at the same time and can switch between them. A user can run multiple copies of the same application at the same time.

***write-ahead log (WAL)***—A transaction logging technique where transactions are recorded in a disk-based log before they are recorded in the physical database. This ensures that active transactions can be rolled back if there is a system crash.



# Index

---

## Symbols

- #alpha 10-5
- #firstbyte 10-5
- #identifier 10-5
- #lower 10-6
- #numeric 10-6
- #prefix 10-7
- #punctuation 10-7
- #secondbyte 10-7
- #translate 10-8
- #upper 10-9
- #whitespace 10-9
- .lcf extension 6-12

## A

- ABORTING transaction status 9-14

- account

  - see also user name, password

  - creating for user 7-3—7-4

- allocating

  - lock entries 3-32

- ALTER DBSECURITY 7-24

- ALTER EXPORTKEY 7-24

- ALTER PASSWORD 7-4

- ALTER TRIGGER

  - LOAD commands 6-9

- anonymous pipes 2-14

  - comdll statement 2-8

  - library name 2-8

  - parameters 2-8

  - sql.ini section 2-8

- ANSI

  - outer join 3-39

- ansijoinsyntax 3-11

- application developers 1-xviii

- archive

  - media 8-6

- area

  - see database area

- AREAS table

  - stored commands A-30

- ASCII

  - character set 3-13

- attribute

  - terminology 1-4

- audit

  - accessing file 4-19

  - category 4-19—4-20

  - clientname field 4-22

  - create database 4-23

  - database field 4-22

  - datetime field 4-22

  - deadlocks and timeouts record 4-24

  - deinstall database 4-23

  - deleting old files 4-19

  - DML record 4-23

  - drop database 4-23

  - elapsedtime field 4-22

  - end of transaction 4-25

  - examples of audit file entries 4-22

  - file name format 4-19

  - file size 4-19

  - global 4-20

  - global type categories table 4-21

  - including message 4-25

  - install database 4-23

  - keeping old files 4-19

  - naming 4-19

  - number of files allowed 4-19

  - number of operations allowed 4-19

  - performance 4-20, 4-21

  - performance of connects and disconnects 4-25

  - performance type categories table 4-22

  - recovery operation 4-23

  - rejected logon 4-23

  - SQL command performance 4-25

  - SQL security violation 4-23

  - start and stop entry examples 4-23

  - start global record 4-23

  - start performance record 4-23, 4-25

  - starting and stopping 4-20

  - stop audit record 4-23

  - table access information 4-25

  - table update information 4-25

  - types 4-20

  - valid connects/disconnects 4-23

  - valid logon/logoff 4-23

- audit file 4-19

- audit keyword

  - described 3-11

- audit operation 4-19

- audit record

  - format 4-22

  - global operations 4-23

  - performance categories 4-25

- authority

- see database authority
- database user levels A-20
- authorization-id 7-9
- AUTOCOMMIT
  - distributed transaction 9-6
- autostartserverpath 3-11

## B

- backend 1-13
- BACKUP 8-3
  - DATABASE 4-24, 8-4
  - LOGBACKUP 8-8
  - LOGS 4-24, 8-4
  - LOGBACKUP 8-8
  - SNAPSHOT 4-24, 8-2, 8-4
    - example 8-12
    - LOGBACKUP 8-4, 8-8
- backup
  - audit record 4-24
  - database 8-3
    - filename 8-12
    - offline 8-3
    - online 8-3
    - partitioned 8-16
  - directory 8-5
  - examples 8-12
  - incremental 8-5
  - MAIN database 4-18, 8-17
  - media
    - archive 8-6
  - offline backups
    - SET NEXTLOG 8-7
  - plan 8-2
  - procedure
    - recommended 8-12
  - restoring offline 8-11
    - SET NEXTLOG 8-7
  - restoring online 8-8
  - schedule
    - guidelines 8-2
- base table 1-8
- batchpriority 3-11
- bindery
  - setting emulation 11-9
- busy indicator
  - NetWare Server Status 5-5

## C

- cache
  - database 3-11, 4-5
  - sort pages 3-49
- cache keyword
  - described 3-11
- catalog
  - see system catalog
- centurydefaultmode keyword
  - described 3-12
- CFL 5-7
- character
  - ASCII set 3-13
- character set keyword
  - described 3-13
- CHECK 6-22
  - DATABASE 6-23
  - SYSTEM ONLY 6-23
  - INDEX 6-23
  - TABLE 6-23
  - WITHOUT INDEXES 6-23
- checking
  - database integrity 6-22
- checkpoint
  - crash recovery 4-6
  - default 4-6, 8-8
  - fuzzy 4-6
  - recovery 8-7
  - time interval 4-6, 8-8
- checksum
  - setting system for 3-36
- classes
  - SQLBase 11-7
- client 1-11
  - components 2-2
  - configuring for NDS 11-10
  - node 3-14
  - programs 5-2
  - stored commands A-31
- Client Node
  - NetWare Server Status field 5-5
- client node (SQLBase Server Console) 5-17
- clientname audit file field 4-22
- clientname keyword
  - described 3-14
- clientruntimedir 3-15
- client-server 1-11
  - advantages 1-12

---

- client-to-server
  - communication 2-4
- cmdtimeout keyword
  - described 3-15
- code page 10-3
  - default 10-3
- column 1-6
  - foreign key A-8
  - index A-11
  - primary key A-14
  - terminology 1-4
- COM+ 2-19
- comdll keyword 2-6
  - described 3-15
- command
  - stored A-5
  - stored for client A-31
  - timeout 3-15, 3-17
- command line
  - keywords 3-4
- comment
  - sql.ini 3-5
- COMMIT 8-7
- commit 3-29, 4-6
  - two-phase 3-17
- commit server 3-17, 9-3, 9-7, 9-14
  - multiple 9-8
- commitserver keyword 9-3, 9-8
  - described 3-17
- COMMITTING transaction status 9-14
- communication
  - client-to-server 2-4
  - libraries 2-2, 3-15
  - matrix 2-4
  - sections for libraries 3-3
- compile
  - procedure 1-9
  - SQL command 1-9
- concurrency 1-10
- configuration file (sql.ini)
  - see Chapter 3
- configuration files 6-24
- CONNECT
  - performance information 4-25
- connect
  - to MAIN database 4-18
- CONNECT authority 7-2
  - granting 7-3
  - revoking 7-5
- connecting
  - to server 6-5, 7-16
- connection
  - local 2-3, 2-14
  - remote 2-3
- Connectivity Administrator 2-5, 2-8, 7-29
- connectpauseticks 3-17
- connecttimeout 3-17
- consistency 1-10
  - ensuring data 4-7
- constraints
  - table A-15
- contents
  - database 1-7
- control file 6-10
  - creating 6-11
  - DIR 6-12, 8-5, 8-6
  - FILEPREFIX 6-11
  - load 6-10
  - SIZE 6-12
  - unload 6-10
- cooperative processing 1-12
- coordinator 9-7
- country 3-6
- country keyword
  - described 3-18, 10-2
- country.dbs
  - NLS 10-4
- country.sql 3-18
  - directives 10-5
  - directives format 10-3
  - location 6-2
  - NLS 10-3
- country.tlk
  - NLS 10-4
- crash recovery 4-7, 8-2
  - checkpointing 4-6
- CRC 7-23
- CREATE
  - DATABASE 6-5, 6-18
    - audit record 4-23
  - DBAREA 6-18
  - SYNONYM 7-9
  - VIEW 7-12
- creating
  - control file 6-11
  - database 6-5, 6-18

- database area 6-18
- database with SQLTalk 6-5
- log file 4-8
- read-only database
  - read-only database
    - creating 6-16
- storage group 6-18
- synonym 7-9
- user account 7-3–7-4
- view 7-12
- CSVPARTS A-24, A-25, A-26, A-27
- CSVTRANS A-24, A-25
- cyclic redundancy check 7-23
- D**
- data
  - how SQL organizes 1-6
  - integrity 1-10
  - preventing loss 8-2
  - recovery 8-8
  - recovery with rollforward 8-9
  - representing relationships 1-3
  - sequence 1-5
  - store 1-10
  - storing in relational database 1-6
  - uniqueness 1-7
- data consistency
  - ensuring 4-7
- data dictionary 1-10
  - see system catalog
- data set
  - terminology 1-4
- DATABASE
  - CHECK 6-23
- Database
  - NetWare Server Status field 5-5
- database 1-6
  - see also distributed database, partitioned database, relational database, replicate database, server, database area
  - adding area 6-19
  - authority
    - RESOURCE 7-2
  - backup 8-3
    - examples 8-12
    - naming backup files 8-12
    - offline 8-3
    - online 8-3
  - cache 3-11, 4-5
  - changing area size 6-19
  - changing structure 1-8
  - contents 1-7
  - creating 6-5, 6-18
  - creating with SQLTalk 6-5
  - dbname keyword 3-20
  - default name 3-23
  - definition 1-7
  - defragmenting 6-21
  - deinstalling 6-6
  - deleting 6-6
  - deleting area 6-19
  - deleting with SQLTalk 6-6
  - directories 3-19
  - directory 4-4
  - distributed 1-13
  - dropping area 6-19
  - encryption 7-20
  - files 4-7
  - fragmentation 6-21
  - hierarchical 1-6
  - integrity 6-22
  - loading 6-8
  - local 2-2
  - MAIN
    - backup 4-18, 8-17
    - database
      - connect 4-18
    - delete 8-17
    - displaying information on 3-49
    - guest 4-18
    - initialization file 4-18
    - partitions 4-15
    - restore 8-17
    - system catalog tables A-24
  - maintenance 1-xviii
  - models
    - comparison 1-6
  - name 4-2
    - default 4-3
  - network
    - model 1-6
  - page alteration protection 7-22
  - partitioned 6-19
    - backup 8-16
    - restore 8-16
  - partitioning 6-17

---

- read-only 6-16
- record messages 3-37
- re-installing 6-8
- relational
  - advantages 1-5
- remote 2-2
- reorganizing 6-21
- restore 8-8
  - examples 8-12
- server name 3-46
- size 4-3
- SQLBase Server Console 5-17
- storage group 6-18
- tables A-16
- template 4-2
- template file 6-5
- unloading 6-8
- views A-16, A-20
- Database Administrator
  - see DBA
- database area
  - adding to/from storage group 6-19
  - creating 6-18
  - deleting 6-19
  - described 4-15, 4-16
  - dropping to/from storage group 6-19
- database audit file field 4-22
- database authority
  - CONNECT 7-2
  - DBA 1-3, 7-2
  - described 7-2
  - revoking 7-5
  - SYSADM 1-3, 7-2
  - SYSADM.SYSUSERAUTH A-20
  - user authority levels A-20
- database integrity
  - checking 6-22
- Databases display 5-5
- DATABASES table
  - stored commands A-29
- datetime audit field 4-22
- DBA 1-3
  - database authority 7-2
  - granting authority 7-3
  - interfaces for tasks 5-2
  - responsibilities 1-2, 1-3
  - revoking authority 7-5
- dbdfault section 3-4
- dbdir keyword 4-4, 6-4
  - described 3-19
- dbname keyword 3-20, 4-2
  - described 3-20
- dbntrsvr.gui 5-21
- DBS.LCF 6-12
- dbwin keyword 5-21
  - described 3-22
- default
  - checkpoint 4-6, 8-8
  - code page 10-3
  - database name 3-23, 4-3
  - LOGBACKUP 8-7
  - password 3-23, 4-3
  - recovery 8-7
  - rollforward 8-10
  - user name 3-23, 4-3
- defaultdatabase keyword 4-3
  - described 3-23
- defaultpassword keyword 4-3
  - described 3-23
- defaultuser keyword 4-3
  - described 3-23
- defaultwrite keyword
  - described 3-23
- defragmenting
  - databases 6-21
  - tables 6-21
- DEINSTALL DATABASE 6-7, 8-7, 8-8
  - audit record 4-23
- deinstalling
  - database 6-6
- declare A-32
- delasa A-32
- delayed password validation 7-27
- deldbs A-32
- deldfl A-32
- deleting
  - database 6-6
  - database area 6-19
  - database with SQLTalk 6-6
  - log files 4-11, 8-3
  - MAIN database 8-17
- delfex A-32
- delfile A-30
- delfxs A-33
- delsta A-32
- delstg A-32

- DIR
    - for control file 6-12, 8-5, 8-6
  - directives
    - #alpha 10-5
    - #firstbyte 10-5
    - #identifier 10-5
    - #lower 10-6
    - #numeric 10-6
    - #prefix 10-7
    - #punctuation 10-7
    - #secondbyte 10-7
    - #translate 10-8
    - #upper 10-9
    - #whitespace 10-9
    - code page 10-3
    - country.sql 10-5
    - format 10-3
  - directory
    - backup 8-5
    - database 3-19, 4-4
    - log file 4-8
  - directsap keyword
    - described 3-25
  - disablelogspacecheck keyword
    - described 3-25
  - DISCONNECT
    - performance information 4-25
  - disconnect
    - last (with distributed transaction) 9-6
  - disk
    - partitions 4-15
    - space 4-15
  - display
    - MAIN database information 3-49
    - multi-user server 3-49
  - displevel keyword 5-21
    - described 3-26
  - DISTRANS 9-4
  - distributed database 1-13
    - advantages 1-15
  - distributed processing 1-11
  - distributed processing network 1-11
  - distributed transaction 9-1—9-15
    - see also two-phase commit
    - AUTOCOMMIT 9-6
    - commit server 9-3, 9-7, 9-14
    - commitserver 9-8
    - commitserver keyword 3-17
    - coordinator 9-7
    - definition 9-2
    - DISTRANS option 9-4
    - example 9-2
    - failure recovery 9-13
    - force resolution 9-14
    - in-doubt 9-7, 9-13, A-13
    - isolation level 9-5
    - last disconnect 9-6
    - locate in SYSADM.SYSPARTTRANS 9-13
    - lock wait timeout 9-5
    - manual recovery 9-13
    - participant 9-7
    - performance issues 9-15
    - savepoint 9-5
    - set up 9-3
      - with SQL/API 9-5
      - with SQLTalk example 9-4
    - status 9-14
    - SYSADM.CSVPARTS 9-8
    - SYSADM.CSVTRANS 9-8, 9-14
    - SYSADM.SYSPARTTRANS 9-7, 9-14
    - system catalog tables A-24
    - with server connects 9-3
  - domain
    - terminology 1-4
  - DROP
    - DATABASE 6-6
      - audit record 4-23
    - SYNONYM 7-11
    - VIEW 7-14
  - duplicate rows 1-7
    - removing 1-5
- ## E
- elapsedtime audit file field 4-22
  - encryption
    - database pages 7-20
    - transaction logs 7-22
    - transmission security 7-26
  - environment variables
    - dbdir 4-4
    - PATH 3-2
    - tempdir keyword 6-16
  - error logging
    - during load operation 6-14
  - error.sql
    - location 6-2

---

- NLS 10-4
- errorfile keyword
  - described 3-26
- event A-6
- EXECUTE
  - RECOMPILE B-2
  - SYSADM.SYSPROC\_ALTTabTRIG B-3
- extdll keyword
  - described 3-27
- extension unit 4-16
- extents
  - described 6-20
  - setting size 6-20
- EXTENTS table
  - stored commands A-29
- external functions 1-6

## **F**

- F1 key
  - SQLBase Server for NetWare 5-4
- F2 key
  - SQLBase Server for NetWare 5-6
- F3 key 5-8
- failure
  - power 8-2
  - recovering distributed transactions 9-13
- FCD 5-7
- field
  - terminology 1-4
- file
  - control 6-10
  - database 4-7
  - general-use 4-14
  - server 1-14
  - sort 4-13
  - temporary 4-13, 6-22
  - terminology 1-4
- file segments
  - limiting size 6-13
  - load/unload operation 6-10
  - specifying directory 6-12
- fileaccess keyword 7-28
  - described 3-28
- FILEPREFIX
  - for control file 6-11
- files
  - remote file protection 7-27
- foreign key

- columns A-8
  - definition 1-7
- format
  - configuration keywords 3-4
- fragmentation
  - database performance 6-21
- free space
  - partitioned databases 6-21
  - querying partitioned databases 6-21
- FREEEXTS table
  - stored commands A-30
- frontend 1-13
- function keys
  - F1 (NetWare Server Status display) 5-4
  - F2 (NetWare Process Activity display) 5-6
  - F3 (NetWare System Activity display) 5-8
  - SQLBase Server for NetWare 5-5, 5-6, 5-8
- fuzzy checkpointing 4-6

## **G**

- general-use files 4-14
- global audit 4-20
- GRANT 7-3, 7-6
  - ALL 7-7
  - ALTER 7-6, 7-7
  - DELETE 7-6
  - INDEX 7-6, 7-7
  - INSERT 7-6
  - SELECT 7-6
  - UPDATE 7-6
- Greenwich Mean Time 3-53
- groupcommit keyword
  - described 3-29
- groupcommitdelay keyword
  - described 3-29
- guest
  - account 4-18
- guidelines
  - scheduling backups 8-2

## **H**

- hierarchical
  - database 1-6
- history file
  - size 4-13
- HP OpenView program 2-17

**I**

- IDLE transaction status 9-14
- incremental backup 8-5
- index 1-6
  - advantages 1-8
  - CHECK INDEX 6-23
  - column A-11
  - definition 1-8
  - integrity check 6-23
  - SYSADM.SYSINDEXES A-9
- in-doubt transaction 9-7, 9-13
  - distributed A-13
- inmessage keyword
  - described 3-30
- input message buffer
  - size 3-30
- insdbs A-32
- insdfl A-32
- insertioncontext keyword
  - described 3-31
  - specifying 11-2
- insext A-30
- inssta A-32
- insstg A-32
- INSTALL DATABASE 6-8, 8-8
  - audit record 4-23
- integrity
  - data 1-10
  - referential 1-7
- integrity check 6-22
  - error 6-23
  - index 6-23
  - system indexes 6-23
  - system tables 6-23
  - table 6-23
- integrity violation 6-23
- interface
  - client programs 5-2
- Internet 2-14
- internet 2-14
- interprocess communication 2-2
- intranet 2-14
- IPX 2-15
- isolation level
  - distributed transaction 9-5
  - read-only 3-44

**J**

- join
  - definition 1-5
  - SELECT 1-5

**K**

- KEEP clause 4-19
- key
  - foreign 1-7
  - foreign columns A-8
  - primary 1-7
  - primary columns A-14
- keywords
  - command line 3-4
  - format 3-4
  - list 3-6

**L**

- LAN (Local Area Network) 1-11
- last disconnect
  - distributed transaction 9-6
- libraries
  - comdll keyword 3-15
- link
  - tables 1-3
- listenport keyword
  - described 3-31
- listenretry keyword
  - described 3-32
- LOAD 6-9
  - control file 6-11
  - error logging 6-14
  - performance 6-14
- loading
  - database 6-8
  - segmented external file 6-10
- Local Area Network (LAN) 1-11
- local connection 2-3, 2-14
- local database 2-2
- location
  - temporary files 3-50
- lock
  - allocating entries 3-32
  - pages 1-10
  - timeout 3-32
- lock wait timeout
  - distributed transaction 9-5
- locks

---

- NetWare System Activity display 5-10
- SQLBase Server Console 5-19

- locks keyword
  - described 3-32

- locktimeout keyword
  - described 3-32

- log
  - NetWare Process Activity 5-7

- log file
  - see also transaction log file
  - checking for space availability 4-9
  - creating 4-8
  - deleting 4-11, 8-3
  - directory 4-8
  - growth 4-9
  - missing 8-11
  - naming 4-8
  - pinned 4-11
  - preallocating 4-9
  - preallocating for growth 3-34
  - Process Activity display 3-32
  - releasing 4-12
  - size 4-9
  - space availability checking 4-9
  - transaction 4-7

- log keyword
  - described 3-32

- LOGBACKUP 4-10, 4-12, 8-3
  - BACKUP

- DATABASE 8-8

- LOGS 8-8

- SNAPSHOT 8-4, 8-8

- default 8-7

- logdir keyword
  - described 3-33

- logfileprealloc keyword
  - described 3-34

- loss
  - preventing for data 8-2

## M

- MAIN database
  - backup 4-18, 8-17
  - delete 8-17
  - display information on 3-49
  - guest 4-18
  - initialization file 4-18
  - partitions 4-15

- restore 8-17

- SYSADM.CSVPARTS 9-8

- SYSADM.CSVTRANS 9-8

- system catalog tables A-24

- MAIN.INI 4-18

- main.ini
  - location 6-2

- mainwin keyword 5-21
  - described 3-34

- Management Information Base (MIB) 2-16
- manual

- organization 1-xviii

- manual recovery
  - distributed transactions 9-13

- maxnestinglevel keyword
  - described 3-35

- media
  - archive 8-6
  - recovery 4-7, 8-2

- memory
  - maximum limitation 3-54

- message types
  - NT Application Event log 12-5

- message.sql
  - location 6-2
  - NLS 10-4

- messages
  - NetWare Process Activity 5-7

- MIB 2-16

## N

- name

- database 4-2

- database backup 8-12

- default user 3-23

- log file 4-8

- object for NDS 11-3

- servername keyword 3-46

- National Language Support 3-6

- national language support
  - see NLS

- NDS 11-2

- advertising SQLBase server 11-1—11-9

- bindery emulation 11-9

- classes 11-7

- configuring clients 11-10

- described 11-1

- setting bindery emulation 11-9

- SQLBase classes 11-7
- SQLBase schema extension 11-6
- ndsloginid keyword
  - described 3-35
- ndsloginpassword keyword
  - described 3-36
- negotiateapi keyword 7-27
- netcheck keyword
  - described 3-36
- netchecktype keyword
  - described 3-37
- netlog keyword
  - described 3-37
- NetWare 11-1
  - client software
    - Microsoft 2-15
    - Novell 2-15
  - communication libraries 2-7
  - name service 3-42
  - starting SQLBase Server 6-2
  - supported protocols 2-4
- NetWare Directory Services
  - see NDS
- NetWare Server Status
  - busy indicator 5-5
  - refresh 5-5
- network
  - database model 1-6
  - purpose 1-11
  - software 2-3
- Network Management Station (NMS) 2-16, 2-17
- NLS
  - configuration files
  - country keyword 10-2
  - country.dbs 10-4
  - country.sql 10-3
  - country.tlk 10-4
  - error.sql 10-4
  - message.sql 10-4
  - sql.ini 10-2
- NMS 2-16, 2-17
- node
  - client 3-14
- Novell Netware 2-15
- NT Application Event Log 12-5
  - event type 12-8
  - logging event types 12-6
  - message events 12-8
  - message types 12-5
- SQLBase level 12-8
- NT service programs
  - administering 12-3
  - overview 12-2
  - service dialog 12-4
  - shutting down 12-8
- nwadvertisemode keyword 11-5
  - described 3-38

## O

- objects
  - SQL 1-6
- ODBC
  - defined
- offline backups 8-3
  - benefits 8-6
  - restoring 8-11
- OLE DB Data Provider 9-6
- online backups 8-3
  - benefits 8-3
  - restoring 8-8
  - segmented 8-5
- Open DataBase Connectivity
  - see ODBC
- operations
  - relational 1-5
- optimizefirstfetch keyword
  - described 3-38
- optimizer
  - setting techniques 3-39
- optimizerlevel keyword
  - described 3-39
- oracleouterjoin keyword
  - described 3-39
- organization
  - of this manual 1-xviii
- organizing
  - data with SQL 1-6
- osavgwindow keyword
  - described 3-40
- ossamplerate keyword
  - described 3-40
- outer join
  - ANSI 3-39
  - Oracle 3-39
- outmessage keyword
  - described 3-41

---

output message buffer  
size 3-41

## **P**

page  
definition 4-6  
locks 1-10  
page alteration protection 7-22  
participant 9-7  
partitioned database  
accessing 3-9, 3-42  
backup 8-16  
free space 6-21  
restore 8-16  
system catalog tables A-24  
partitioning 4-15  
database 6-17  
MAIN database 4-15  
partitions keyword  
described 3-42  
password  
changing user 7-4  
default 3-23, 4-3  
delayed validation 7-27  
NDS 11-5  
server connection 3-42, 7-16  
server security 3-42, 7-17  
password keyword  
described 3-42  
PATH environment variable 3-2  
PC (Personal Computer) 1-11  
performance audit 4-20  
personal computer (PC) 1-11  
pinned log files 4-11  
plan  
backup 8-2  
platforms  
names 2-6  
supported protocols 2-4  
PNM 5-7  
power  
failure 8-2  
preallocating  
log file 4-9  
transaction log file 3-34  
preferrednameservice keyword 11-10  
described 3-42  
PREPARED transaction status 9-14

preventing  
loss of data 8-2  
primary key  
columns A-14  
definition 1-7  
privileges  
revoking 7-8  
table 7-6, A-15  
updating for user A-3  
view 7-6  
procedure  
compile 1-9  
description 1-9  
store 1-9  
Process Activity  
display log 3-32  
processing  
cooperative 1-12  
distributed 1-11  
procwin keyword 5-21  
described 3-43, 3-46, 7-26, 7-27  
programs  
client 5-2  
projection  
definition 1-5  
SELECT 1-5  
protocol  
supported platforms 2-4  
two-phase commit 3-17  
PUBLIC 7-7, 7-8  
public  
synonyms 7-10

## **R**

read-only  
history file 4-13  
isolation level 3-44  
transactions 4-13  
read-only database  
benefits 6-16  
displaying setting 6-16  
SET  
READONLYDATABASE 6-16  
transaction log file 6-16  
readonly keyword  
described 3-44  
RECOMPILE  
EXECUTE B-2

- recompile stored commands B-2
- record
  - database messages 3-37
  - terminology 1-4
- recovery
  - automatic 9-13
  - commands 8-7
  - crash 4-7
  - default 8-7
  - disable 4-12
  - manual 9-13
  - media 4-7, 8-2
  - of data 8-8
  - off 8-7
  - rollforward 8-9
- recovery operation audit record 4-23
- redirect
  - transaction logs 3-33
- referential integrity 1-7
- Registry 6-24
- registry
  - primary configuration information 3-54
  - running multiple versions 6-24
  - service configuration information 12-4
  - service programs 12-4
- re-installing
  - database 6-8
- relation
  - terminology 1-4
- relational database
  - advantages 1-3, 1-5
  - definition 1-3
  - how it works 1-4
  - store data 1-6
- relational operations 1-5
- relationships 1-3
- release
  - log file 4-12
- RELEASE LOG 4-24
- remote connection 2-3
- remote database 2-2
  - SQLConsole server tool 5-3
- remote file protection 7-27
- REORGANIZE 6-21
  - with encrypted databases 7-22
- reorganizing
  - database 6-21
- RESOURCE
  - authority 7-2
    - granting 7-3
    - revoking 7-5
  - responsibilities
    - DBA 1-3
    - SYSADM 1-3
  - RESTORE 8-8
    - DATABASE 4-24, 8-9, 8-11
    - LOGS 8-11
    - SNAPSHOT 4-24, 8-9
      - example 8-12
  - restoring
    - backup audit record 4-24
    - database 8-8
    - database examples 8-12
    - MAIN database 8-17
    - offline backups 8-11
    - online backups 8-8
    - partitioned database 8-16
    - procedure for database 8-12
  - result set 1-4
  - retry keyword
    - described 3-17
  - retrytimeout keyword
    - described 3-44
  - return codes
    - user-defined 3-26
  - REVOKE 7-5, 7-8
    - ALTER 7-8
    - CONNECT authority 7-5
    - database authority 7-5
    - INDEX 7-8
    - privileges 7-8
    - RESOURCE authority 7-5
  - ROLLBACK 8-7
  - rollback 4-7
  - ROLLBACK TRANSACTION FORCE 9-14
  - ROLLFORWARD 4-24, 8-10, 8-11
    - CONTINUE 8-11
    - TO BACKUP 8-10
    - TO END 8-10
    - TO TIME 8-10
  - rollforward 8-9
    - default 8-10
  - row
    - duplicates 1-7
    - removing duplicate 1-5
    - retrieving 1-5

---

- terminology 1-4
- rules
- syntax configuration 3-4

## S

- savepoint
  - distributed transaction 9-5
- schedule
  - backup guidelines 8-2
- searchcontext keyword 11-11
  - described 3-44
- secure hash method 7-23
- secureapi keyword 7-26
  - described 3-46
- security
  - database page alteration protection 7-22
  - database page encryption 7-20
  - delayed password validation 7-27
  - remote file protection 7-27
  - server 7-16
  - server connection password 7-16
  - server security password 7-17
  - system catalog 7-15
  - transaction log encryption 7-22
  - transmission 7-25
- segmenting
  - load/unload external file 6-10
- seladb A-32
- selare A-32
- seldb A-29
- seldbs A-32
- seldfl A-32
- seldfs A-32
- SELECT
  - join 1-5
  - projection 1-5
- selection
  - definition 1-5
- selexa A-31
- selexf A-32
- selexta A-29
- selexts A-29
- selgt A-30
- selleg A-30
- sellog A-29
- selname A-29
- selsgd A-32
- selsgp A-32
- selsize A-29
- server 1-11
  - attributes 1-12
  - commit 3-17
  - components 2-2
  - configuring on start up 6-4
  - connecting 7-16
  - connecting to 6-5
  - connection password 3-42, 7-16
  - criteria 1-12
  - file 1-14
  - security 7-16
  - security password 3-42, 7-17
  - servername keyword 3-46
  - setting multi-user display 3-49
  - starting for NetWare 6-2
  - starting for Windows 95/98 6-3
  - starting for Windows NT 6-3
  - stopping for NetWare 6-3
  - stopping for Windows 95/98 6-4
  - stopping for Windows NT 6-4
- Server Status display
  - see SQLBase Server for NetWare, SQLBase Server Monitor
- servernames keyword
  - described 3-47
- serverpath keyword
  - described 3-47
- SET
  - BINDERY CONTEXT 11-9
  - CHECKPOINT 8-8
  - EXTENSION 4-16, 6-20
  - LOGBACKUP 8-7
  - NEXTLOG 8-7
  - PARTITIONS 8-17
  - PRINTLEVEL 5-15
  - READONLYDATABASE
    - transaction log file 6-16
  - RECOVERY 8-7
  - SERVER 7-17
  - SPANLIMIT 4-10
  - TIMESTAMP 5-15
- set
  - definition 1-4
- set theory 1-4
- SHA 7-23
- showmaindbname keyword
  - described 3-49

- silentmode keyword
  - described 3-49
- SIZE
  - for control file 6-12
  - parameter 4-3
- SMC 5-21
- SNAPSHOT
  - BACKUP
    - LOGBACKUP 8-4
- SNM 5-7
- SNMP 2-15
  - agent 2-16
  - components 2-16
- sockets 2-15
- sort
  - cache pages 3-49
- sort file 4-13
- sortcache keyword
  - described 3-49
- space
  - allocation unit 3-53
  - checking for log file 4-9
- span limit
  - transaction 4-10
- SPX 2-15
  - comdll statement 2-8
  - library name 2-8
  - parameters 2-8
  - sql.ini section 2-8
- SQL 1-14
  - capabilities 5-3
  - command audit record 4-25
  - limitations 5-3
  - objects 1-6
  - organizing data with 1-6
- sql.ini 3-2, 4-3
  - comment 3-5
  - configuring communications 2-5
  - dnntsrv.gui 5-21
  - editing 3-3
  - entries 3-4
  - keyword format 3-4
  - keyword list 3-6
  - locating 3-2
  - location 6-2
  - NLS 10-2
  - sections 3-3
  - setting keywords at startup 6-4
  - specifying keywords on command line 6-4
  - structure 2-5
  - syntax rules 3-4
  - values 3-4
- SQL/API 1-xviii
  - using 5-3
  - with distributed transactions 9-5
- sqlapipe.dll 2-8
- SQLBase
  - multiple versions 6-23
  - start 6-2
  - stop 6-2
- SQLBase Management Console 5-21, 6-24
- SQLBase Resouce Manager 5-23
- SQLBase Resource Manager 2-19, 5-25, 6-24
- SQLBase Server Console 5-12
  - configuring 5-13, 5-21
  - Databases window 5-17
  - Display menu options 5-15
  - File menu 5-14
  - Level menu 5-15
  - Pause menu 5-15
  - pausing 5-13
  - Process Activity window 5-17
  - scrolling 5-13
  - Security menu 5-16
  - Server Status window 5-17
  - System Activity window 5-18
  - Window menu 5-16
- SQLBase Server for NetWare 5-8
  - advertising on NDS 11-1—11-9
  - character-based server interface 5-4—5-12
  - function keys 5-8
  - NDS schema extension 11-6
  - Process Activity display
    - Database process information fields 5-6
    - log 5-7
    - message levels 5-7
  - server function keys 5-5, 5-6
  - Server Status display fields 5-5
  - Server Status display screen 5-4
  - System Activity display
    - Databases fields 5-10
    - Process information fields 5-9
    - System data fields 5-9
- SQLBase Server for Windows 95 5-12
- SQLBase Server for Windows NT
  - installing 12-2

---

- running as an application program 12-8
- running as an NT service 12-1, 12-2
- shutting down 12-8
- SQLBase Server Monitor 5-26
  - Help menu 5-16
  - switching installation 6-25
- SQLBASE::Database class 11-8
- SQLBASE::DBServer class 11-8
- sqlbdb 8-4
- sqlblf 8-4
- SQLBRM 2-19
- sqlbss 8-4
- sqlcnr 4-12
- SQLConsole
  - described 5-3
- sqlcre 6-5
- sqlcrf 8-11
- sqlded 6-7, 8-7, 8-8
- sqldel 6-6
- sqlenr 8-11
- sqlfgt
  - preventing calls 7-27
- sqlfpt
  - preventing calls 7-27
- sqlind 6-8, 8-8
- sqlldb 6-9
- sqlmdl
  - preventing calls 7-28
- sqlmop
  - preventing calls 7-27
- SQLMPIPE 2-20
- SQLPCTI 8-8
- SQLPDTR 9-5
- SQLPEXS 4-16
- SQLPLBM 8-7
- sqlrdb 8-9, 8-11
- sqlrof 8-10
- sqlrss 8-8, 8-9
- sqlsp32.dll 2-8
- SQLTalk 1-xviii
  - definition 5-2
  - session 5-2
- sqlunl 6-9
- sqlws32.dll 2-7
- sqlwsspx.dll 2-8
- stack
  - setting size 3-52
- start.dbs 4-2, 6-5
- starting
  - SQLBase 6-2
- Status
  - NetWare Server Status field 5-5
  - SQLBase Server Console 5-17
- statwin keyword 5-21
  - described 3-50
- stopping
  - SQLBase 6-2
- storage group
  - adding/dropping database area 6-19
  - changing 6-20
  - creating 6-18
  - default setting 6-20
  - deleting 6-20
  - described 6-20
  - dropping database area 6-19
  - explanation 4-17
- STORE
  - procedure 1-9
  - SQL command 1-9
- store
  - data 1-10
  - view 1-8
- stored
  - client commands A-31
- stored commands 1-6
  - MAIN database A-29
  - SYSADM.SYSCOMMANDS A-5
- stored procedures 1-6
  - Centura-supplied B-2
  - execute privileges 7-14
  - RECOMPILE B-2
  - SYSADM.SYSCOMMANDS A-5
  - SYSADM.SYSPROC\_ALTTABTRIG B-3
- structure
  - changing for database 1-8
- synonym 1-6
  - creating 7-9
  - definition 1-9, 7-9
  - described 7-9
  - public 7-10
  - table A-14
  - view A-14
- syntax
  - configuration rules 3-4
- SYSADM 1-3, A-20
  - database authority 7-2

- granting database authority 7-3
- responsibilities 1-3
- system catalog privileges 7-15
- SYSADM.CSVPARTS 9-8
- SYSADM.CSVTRANS 9-8, 9-14
- SYSADM.SYSCOLAUTH A-3
- SYSADM.SYSCOLUMNS A-3
- SYSADM.SYSCOMMANDS A-5
- SYSADM.SYSDependencies A-6
- SYSADM.SYSEVENTS A-6
- SYSADM.SYSEXECUTEAUTH A-7
- SYSADM.SYSEXTFUN A-7
- SYSADM.SYSEXTPARAM A-8
- SYSADM.SYSFKCONSTRAINTS A-8
- SYSADM.SYSINDEXES A-9
- SYSADM.SYSKEYS A-11
- SYSADM.SYSOBAUTH A-11
- SYSADM.SYSOBJSYN 7-10, A-12
- SYSADM.SYSPARTTRANS 9-7, 9-14, A-13
- SYSADM.SYSPKCONSTRAINTS A-14
- SYSADM.SYSPROC\_ALTTabTrig B-3
- SYSADM.SYSROWIDLISTS A-14
- SYSADM.SYSSYNONYMS 7-10, A-14
- SYSADM.SYSTABAUTH A-15
- SYSADM.SYSTABCONSTRAINTS A-15
- SYSADM.SYSTABLES A-16
- SYSADM.SYSTRGCOLS A-18
- SYSADM.SYSTRIGGERS A-19
- SYSADM.SYSUSERAUTH A-20
- SYSADM.SYSVIEWS A-20
- SYSCOLAUTH 7-15
- SYSCOMMITORDER C-2
- SYSEXECUTEAUTH 7-15
- SYSOBJAUTH 7-15
- SYSSQL.SYSCOLAUTH A-22
- SYSSQL.SYSCOLUMNS A-22
- SYSSQL.SYSCOMMANDS A-22
- SYSSQL.SYSDependencies A-22
- SYSSQL.SYSEXECUTEAUTH A-23
- SYSSQL.SYSINDEXES A-23
- SYSSQL.SYSKEYS A-23
- SYSSQL.SYSOBAUTH A-23
- SYSSQL.SYSOBJSYN A-23
- SYSSQL.SYSPARTTRANS A-24
- SYSSQL.SYSSYNONYMS A-24
- SYSSQL.SYSTABAUTH A-24
- SYSSQL.SYSTABLES A-24
- SYSTABAUTH 7-15

- System Activity
  - display 5-8
- system administrators 1-xviii
- system catalog 1-10
  - granting access 7-15
  - MAIN database tables A-24
  - non-SQLBase views A-22
  - partitioned database tables A-24
  - security 7-15
  - table operations A-20
  - tables for distributed transactions A-24
  - tables list A-2
  - views A-21
- SYSTEM ONLY
  - CHECK DATABASE 6-23
- SYSTIMEZONE 3-53
- SYSUSERAUTH 7-15
- syswin keyword 5-21
  - described 3-50

## **T**

- TABLE
  - CHECK 6-23
- tables 1-6, A-16
  - AREAS
    - stored commands A-30
  - base 1-8
  - constraints A-15
  - DATABASES
    - stored commands A-29
  - defragmenting 6-21
  - EXTENTS
    - stored commands A-29
  - FREEEXTS
    - stored commands A-30
  - integrity check 6-23
  - link 1-3
  - privileges 7-6, A-15
  - query audit information 4-25
  - synonym A-14
  - terminology 1-4
  - update information for audits 4-25
- TCP/IP 2-14
  - comdll statement 2-7
  - library name 2-7
  - parameters 2-7
  - sql.ini section 2-7
- Team Developer 1-xviii

---

tempdir keyword 4-13, 6-16  
template  
    database 4-2, 6-5  
temporary files 4-13, 6-22  
    location 3-50  
terminology 1-4  
threadmode keyword  
    described 3-51  
threadstacksize keyword  
    described 3-52  
time out 3-52  
timeout  
    command 3-15, 3-17  
    lock 3-32  
timeout keyword  
    described 3-52  
timestamps keyword  
    described 3-52  
timezone keyword  
    described 3-53  
tlidll.nlm 2-7  
tlispx.nlm 2-8  
transaction  
    see also distributed transaction  
    control 4-7, 8-7  
    deleting log files 8-3  
    end-of information 4-25  
    log files 4-7  
    logs  
        encryption 7-22  
    preallocating log file 3-34  
    read-only 4-13  
    span limit 4-10  
transaction logs  
    read-only database 6-16  
    redirect 3-33  
    SET READONLYDATABASE 6-16  
transmission security 7-25  
Treasury Edition 7-21  
triggers 1-6  
    definition 1-10  
    enabling/disabling B-3  
    SYSADM.SYSTRGCOLS A-18  
    SYSADM.SYSTRIGGERS A-19  
    unload operation 6-10  
    update A-18  
tuple  
    terminology 1-4

two-phase commit 9-3, 9-6–9-12  
    commit server 9-3, 9-7  
    components 9-7  
    participant 9-7  
    protocol 3-17  
    SYSADM.SYSTPARTTRANS 9-7

## U

uniqueness  
    data 1-7  
unit  
    space allocation 3-53  
UNLOAD 6-9  
    control file 6-10  
    error logging 6-14  
unloading  
    database 6-8  
    segmented external file 6-10  
update  
    privileges of users A-3  
    trigger A-18  
upddbsd A-32  
upddbsl A-32  
updfxs A-32  
USER 7-16, A-21  
user  
    changing passwords 7-4  
    creating account for 7-3–7-4  
    default name 4-3  
    revoking database authority 7-5  
    table privileges A-15  
User Name  
    NetWare Server Status field 5-5  
    SQLBase Server Console 5-17  
user name  
    default 3-23  
    NDS 11-5  
user privileges  
    update A-3  
users A-20  
    database authority levels A-20  
users keyword  
    described 3-53

## V

view 1-6, A-16, A-20  
    base table 1-8  
    benefits 7-12

- creating 7-12
- definition 1-8
- described 7-12
- non-SQLBase system catalog A-22
- privileges 7-6
- storage 1-8
- synonym A-14
- system catalog A-21
- use 1-8
- WITH CHECK OPTION 7-12

## W

- Windows 95/98
  - communication libraries 2-7
  - starting server 6-3
  - supported protocols 2-4
- Windows NT
  - running SQLBase as a service 12-1
  - starting server 6-3
  - supported protocols 2-4
- Windows NT registry
  - service programs 12-4
- winsock 2-15
- WITH CHECK OPTION 7-12
- WITHOUT INDEXES
  - CHECK TABLE 6-23
- workalloc keyword
  - described 3-53
- worklimit keyword
  - described 3-54

## Y

- Y2K 3-12
- year 2000 3-12



