

SQLBase 9.0 Migratory Path

By Charles McLouth
Product Manager - SQLBase

September 2004

GUPTA™

Table of Contents

Introduction	3
Migrating SQLBase Servers and Databases.....	3
Introduction	3
Migrating Databases – Phase I	4
Installation/Configuration.....	4
Migrating Databases - Phase II.....	5
Special Considerations.....	6
Conclusion.....	6
Client Applications.....	6
Introduction	6
Installing and Configuring the SQLBase 9.0 Client	6
Migrating 32-bit applications.....	6
Migrating 16-bit applications.....	7
Conclusion.....	8

Introduction

SQLBase is GUPTA's core flagship product that delivers the technology required for today's enterprise applications, be they distributed over the Internet or an Intranet, or expected to function in a highly integrated, distributed environment.

With SQLBase, companies can manage data far from the corporate data center with administration-free operation and low resource requirements. Best of all, SQLBase can significantly reduce the overall cost of developing and deploying a remote application employing an embedded database.

Through new features, critical enhancements to tried-and-true technology, and adoption of emerging industry standards, SQLBase 9.0 supports the evolution of your business applications.

SQLBase 9.0 will increase the potential of business applications because it:

- Simplifies deployment through its deployment kit for Independent Software Vendors (ISVs)
- Reduces operational and support costs through proven reliability, low admin efforts, self tuning and self maintaining operations
- Is a fully relational database, guaranteeing data integrity through transaction support and automatic crash recovery
- Provides optimized performance for laptop to workgroup servers
- Provides complete flexibility for client access
- Offers end-to-end security up to triple-DES encryption for full database and on the wire encryption
- Supports the SQL Standard, allowing one application to talk to databases from different vendors
- Provides a rich client toolset with monitoring tools
- Supports business transactions utilizing COM+ transactions (Microsoft Transaction Server - MTS) in a distributed architecture

This paper will discuss the issues and features you need to consider while migrating SQLBase Servers and database from versions 5.x to 9.0.

This paper uses SQLBase 9.0 Server to refer to SQLBase Desktop and Server versions.

Migrating SQLBase Servers and Databases

Introduction

Migrating to SQLBase 9.0 Server, from prior versions (5.x, 6.x, 7.x, & 8.x) is a simple three-step approach:

1. Migrating Databases Phase I
2. Installation/Configuration
3. Migrating Databases Phase II

This section will provide step-by-step instructions to perform all three.

Migrating Databases – Phase I

With few restrictions, SQLBase 9.0 will automatically migrate most databases to current format during the first client connection to the database. The restrictions for automatic migration of databases are:

- The database must be a version 6.0.01 or greater.
- The database must be in a consistent state. This means that it cannot be in a corrupted state or have any log file dependencies.
- The database cannot have any user-created dependencies (like views, indexes, or synonyms) upon SQLBase's system tables or views (see DBA Guide for list of system tables and views.)
- Encrypted databases MUST be the same encryption level as the SQLBase 9.0 Server. A backup (BACKUP DATABASE or BACKUP SNAPSHOT) cannot be "RESTORED" with the SQLBase 9.0 Server.
- Partitioned databases cannot be automatically converted.

If these restrictions do not apply, then proceed to "[Installation/Configuration](#)."

Databases that cannot be automatically migrated to SQLBase 9.0 must first be exported to an ASCII format. This process is called an "Unload" and is performed using the interactive user interface for SQLBase called SQLTalk:

- tlkntr.exe-SQLBase on Netware
- sqlxdemo.exe-SQLBase on Win16
- sqlnttlk.exe, sqlndemo.exe, or sqltalk.exe-SQLBase on Win32

Using SQLTalk, connect to the database as user "SYSADM", and perform the command "UNLOAD DATABASE" (Refer to SQLTalk Command Reference or SQLBase Database Administrator's Guide for more information on the UNLOAD command.)

Example:

```
UNLOAD DATABASE <filename>;  
(where filename is the fully qualified filename of the resulting file.)
```

(note: if the database is very large, then it is recommended that a Segmented Unload be performed. For information about Segmented Unloads, refer to SQLBase Tech Alert "[REORGANIZE and Large Files](#)".) Upon completion of this step, proceed to "[Installation/Configuration](#)." For more information on Database migration, refer to the SQLBase 9.0 Release Notes (<http://www.guptaworldwide.com/Support/Default.aspx>).

Note: *It is strongly recommended that prior to migrating databases, a backup of the system be performed to ensure recovery in the event of a hardware failure.*

Installation/Configuration

SQLBase 9.0 contains a simple and easy to use installation and configuration process. To start the installation process, mount the CD (if the SQLBase installation does not begin launch setup.exe(Windows) or install.sh(Linux).)

Configuring the SQLBase 9.0 Server differs depending upon platform (Windows or Linux), but the only requirements are to enable the

SQLBase Communication DLLs (comdll) for connectivity and specify the location of database files (dbdir.)

Configuring SQLBase for Windows

To configure SQLBase 9.0, use the tool "GUPTA Connectivity Administrator" and set the Database Directory (dbdir) to the desired location and enable communication dlls of TCP/IP(sqlws32) and Local (sqlpipe.)

For databases that can be automatically migrated to SQLBase 9.0 (see [Migrating Databases Phase I](#) for restrictions,) add the database names to the list of serviced databases.

Configuring SQLBase for Linux

To configure SQLBase 9.0, use a text editor of your choosing and in the Server Section [dblxsrv] specify the Database Directory (dbdir) [ensure that this path is from the context of the Linux server.

Example:

```
/opt/gupta/sqlbase90] and specify the TCP/IP communication dll (sqltcpip) in the servename parameter and dbname parameters (ex: servename=server1, sqltcpip & dbname=island, sqltcpip.)
```

For databases that can be automatically migrated to SQLBase 9.0 (see "[Migrating Databases Phase I](#)" for restrictions,) add parameters dbname=<dbname>, sqltcpip for each database desired to be migrated.

Lastly, enable the communication dll in the Server dll Section [dblxsrv.dll], replace the existing comdll entry with comdll= sqltcpip.

Example:

```
[dblxsrv]
servename=server1,sqltcpip
dbname=island,sqltcpip
dbdir=/opt/gupta/sqlbase90
```

```
[dblxsrv.dll]
comdll=sqltcpip
```

Migrating Databases - Phase II

For databases that can be automatically migrated to SQLBase 9.0 (see [Migrating Databases Phase I](#) for restrictions,) copy existing database files and folders to the Database Directory (dbdir) of the SQLBase Server. Start the SQLBase Server (*Windows use SQLBase Management Console and for Linux either invoke "SQLBase Database Engine" from "X" or ./dblxsrv from a shell.*) Using SQLTalk v9 (sqltalk.exe on Windows and sqlxltk on Linux) connect to each database, at which point the databases will be automatically migrated to SQLBase 9.0. For databases that cannot be automatically migrated, start the SQLBase Server (*Windows use SQLBase Management Console and for Linux either invoke "SQLBase Database Engine" from your "X" program group or ./dblxsrv from a shell.*)

Using SQLTalk v9 (sqltalk.exe on Windows and sqlxltxl on Linux), create a new database (see SQLTalk command Reference for "Creating a Database") for each database intended for manual migration. Next connect to each database and import the ASCII file (the "UNLOAD" file created in [Migrating Databases Phase I](#)) with the "LOAD" command (Refer to SQLTalk Command Reference or SQLBase Database Administrator's Guide for more information on the LOAD command.)

Example:

```
LOAD SQL <filename>;  
(where filename is the fully qualified filename of the resulting  
file.
```

Special Considerations

If migrating from Win16, then it is important to note differences in the behavior of 16-bit SQLBase Server and the 32-bit SQLBase Server. The significant differences are that each database being serviced by the SQLBase Server must be added as a dbname parameter to the Server's configuration. This was not required with the 16-bit version of the SQLBase Server.

If migrating from version prior to 6.1.02, then it is important to note that the SQL function @DECRYPT() has been deprecated and no longer is supported.

Conclusion

Upon completing these steps the SQLBase Server and its databases will have been successfully converted to SQLBase Server v9.0.

Client Applications

Introduction

Migrating Client Applications that use SQLBase as a database generally do not require much effort to migrate to SQLBase 9.0. However migrating 16-bit code to 32-bit Windows can be a significant effort. This section explains, the requirements and recommended process for migrating Client Applications to SQLBase 9.0.

Installing and Configuring the SQLBase 9.0 Client

The SQLBase 9.0 CD now contains a "Redistribute-able Components" that will provide an installer for ease of deployment. After installation, you can configure your SQLBase 9.0 Client by using the "GUPTA Connectivity Administrator." Using the "Client" tab, add/replace SQLBase TCP/IP connections to SQLBase Servers (optionally, Local Connection for configurations that have both the client application and SQLBase Server on the same machine.)

Migrating 32-bit applications

There are no coding requirements needed for migrating an existing 32-bit application to SQLBase 9.0. C/API applications are required to be re-built with the newly supplied header and library files. The only requirement is the deployment of the SQLBase 9.0 Redistribute-able components (available from the SQLBase 9.0 installation CD.) (Note:

SQLBase 8.0 introduced a new 3.5 ODBC Driver. The previous version of the ODBC Driver was a 2.x driver. The new driver is functionally compatible with the 2.x driver, but 2.x ODBC applications may observe subtle behavior differences caused by the Microsoft ODBC Driver Manager in its mapping from 3.x to 2.x.)

Migrating 16-bit applications

Although unrelated to SQLBase, there may be significant overhead in migrating a 16-bit Windows application to a 32-bit Windows application. Therefore, it is recommended that it occur in a "Multiple Phased" approach, which allows for a staged deployment approach. This approach is explained below in detail.

Migrating 16-bit applications (Phase I)

The first step of migrating the client application to SQLBase 9.0 is to migrate the SQLBase Server (refer to [Server and Databases](#).)

The next step is to migrate the 16-bit SQLBase Client to the 16-bit SQLBase Client v7.5.01 (*the SQLBase 9.0 Server is backward compatible with 7.5 and 7.6 client software.*) This step is a simple replacement of the 16-bit SQLBase Client files currently deployed with the 16-bit SQLBase 7.5.01 Client files.

Migrating 16-bit applications (Phase II)

The next phase for Migrating 16-bit applications requires code changes to the application and is dependent upon the development model and language of the client application. The below sections are specific to development models of the application.

➤ C/API

C/API applications linked with the SQLBase SQLAPI do not require code changes as relevant to SQLBase, but will require significant code changes. For information about migrating your 16-bit code to 32-bit Windows, refer to Microsoft document "[Porting 16-Bit Code to 32-Bit Windows](#)"

(http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore/html/core_porting_16.2d.bit_code_to_32.2d.bit_windows.asp). It is recommended that the application be migrated on a component-by-component basis, where each new 32-bit component uses the 32-bit SQLBase 9.0 Client and each remaining 16-bit components continue to use the 16-bit SQLBase 7.5.01 Client.

Utilizing the *.h files and sqlwntm.lib, provided by the SQLBase 9.0 Development Components, to build and link there are no additional changes required.

➤ ODBC

The second phase of an ODBC project involves two steps. According to Microsoft document "[Using 16-Bit Applications with 32-Bit Drivers](#)"

(<http://msdn.microsoft.com/library/default.asp?url=/library/en-us/odbc/htm/using16bitapplicationswith32bitdrivers.asp>), it is possible for 16-bit ODBC applications to use the 32-bit SQLBase 9.0 ODBC Driver. After this step, all SQLBase aspects of the client application will be using the 32-bit SQLBase 9.0 features and functionality.

However, it is recommended that the remaining step of migrating the 16-bit code to 32-bit code be taken. (For information about migrating your 16-bit code to 32-bit Windows, refer to Microsoft document "[Porting 16-Bit Code to 32-Bit Windows](#)"

http://msdn.microsoft.com/library/default.asp?url=/library/en-us/vccore/html/core_porting_16.2d.bit_code_to_32.2d.bit_windows.asp.)

➤ SQLWindows

There are no SQLBase specific requirements for migrating 16-bit SQLWindows applications to 32-bit SQLWindows applications. Refer to "Migrating SQLWindows Applications."

Conclusion

While migrating 32-bit applications to SQLBase 9.0, is a trivial effort, migrating 16-bit applications to 32-bit Windows needs to be carefully planned and scheduled. The steps outlined in this document are a guideline that will aid in the planning and implementation of either type of project.

Copyright © 2004 Gupta Technologies, LLC. GUPTA, the GUPTA logo, and all GUPTA products are licensed or registered trademarks of Gupta Technologies, LLC. All other products are trademarks or registered trademarks of their respective owners. All rights reserved.