SYBASE iAnywhere

SQL Anywhere® 11
Introduction

February 2009

Version 11.0.1

Copyright and trademarks

Contents

About this book

Subject

This book introduces SQL Anywhere 11, a comprehensive package that provides data management and data exchange, enabling the rapid development of database-powered applications for server, desktop, mobile, and remote office environments.

Audience

This book is designed to highlight the main features of SQL Anywhere 11 to help application developers and database administrators assess its scope and functionality.

Before you begin

This book assumes some familiarity with relational databases.

About the SQL Anywhere documentation

The complete SQL Anywhere documentation is available in four formats that contain identical information.

- **HTML Help** The online Help contains the complete SQL Anywhere documentation, including the books and the context-sensitive help for SQL Anywhere tools.

  If you are using a Microsoft Windows operating system, the online Help is provided in HTML Help (CHM) format. To access the documentation, choose **Start** » **Programs** » **SQL Anywhere 11** » **Documentation** » **Online Books**.

  The administration tools use the same online documentation for their Help features.

- **Eclipse** On Unix platforms, the complete online Help is provided in Eclipse format. To access the documentation, run *sadoc* from the *bin32* or *bin64* directory of your SQL Anywhere 11 installation.

- **DocCommentXchange** DocCommentXchange is a community for accessing and discussing SQL Anywhere documentation.

  Use DocCommentXchange to:

  - View documentation
  - Check for clarifications users have made to sections of documentation
  - Provide suggestions and corrections to improve documentation for all users in future releases

  Visit http://dcx.sybase.com.

- **PDF** The complete set of SQL Anywhere books is provided as a set of Portable Document Format (PDF) files. You must have a PDF reader to view information. To download Adobe Reader, visit http://get.adobe.com/reader/.

  To access the PDF documentation on Microsoft Windows operating systems, choose **Start** » **Programs** » **SQL Anywhere 11** » **Documentation** » **Online Books - PDF Format**.

  To access the PDF documentation on Unix operating systems, use a web browser to open *install-dir/documentation/en/pdf/index.html*.

About the books in the documentation set

The SQL Anywhere documentation consists of the following books:

- **SQL Anywhere 11 - Introduction** This book introduces SQL Anywhere 11, a comprehensive package that provides data management and data exchange, enabling the rapid development of database-powered applications for server, desktop, mobile, and remote office environments.
- **SQL Anywhere 11 - Changes and Upgrading** This book describes new features in SQL Anywhere 11 and in previous versions of the software.
- **SQL Anywhere Server - Database Administration** This book describes how to run, manage, and configure SQL Anywhere databases. It describes database connections, the database server, database

files, backup procedures, security, high availability, replication with the Replication Server, and administration utilities and options.

- **SQL Anywhere Server - Programming** This book describes how to build and deploy database applications using the C, C++, Java, PHP, Perl, Python, and .NET programming languages such as Visual Basic and Visual C#. A variety of programming interfaces such as ADO.NET and ODBC are described.
- **SQL Anywhere Server - SQL Reference** This book provides reference information for system procedures, and the catalog (system tables and views). It also provides an explanation of the SQL Anywhere implementation of the SQL language (search conditions, syntax, data types, and functions).
- **SQL Anywhere Server - SQL Usage** This book describes how to design and create databases; how to import, export, and modify data; how to retrieve data; and how to build stored procedures and triggers.
- **MobiLink - Getting Started** This book introduces MobiLink, a session-based relational-database synchronization system. MobiLink technology allows two-way replication and is well suited to mobile computing environments.
- **MobiLink - Client Administration** This book describes how to set up, configure, and synchronize MobiLink clients. MobiLink clients can be SQL Anywhere or UltraLite databases. This book also describes the Dbmlsync API, which allows you to integrate synchronization seamlessly into your C++ or .NET client applications.
- **MobiLink - Server Administration** This book describes how to set up and administer MobiLink applications.
- **MobiLink - Server-Initiated Synchronization** This book describes MobiLink server-initiated synchronization, a feature that allows the MobiLink server to initiate synchronization or perform actions on remote devices.
- **QAnywhere** This book describes QAnywhere, which is a messaging platform for mobile, wireless, desktop, and laptop clients.
- **SQL Remote** This book describes the SQL Remote data replication system for mobile computing, which enables sharing of data between a SQL Anywhere consolidated database and many SQL Anywhere remote databases using an indirect link such as email or file transfer.
- **UltraLite - Database Management and Reference** This book introduces the UltraLite database system for small devices.
- **UltraLite - C and C++ Programming** This book describes UltraLite C and C++ programming interfaces. With UltraLite, you can develop and deploy database applications to handheld, mobile, or embedded devices.
- **UltraLite - M-Business Anywhere Programming** This book describes UltraLite for M-Business Anywhere. With UltraLite for M-Business Anywhere you can develop and deploy web-based database applications to handheld, mobile, or embedded devices, running Palm OS, Windows Mobile, or Windows.
- **UltraLite - .NET Programming** This book describes UltraLite.NET. With UltraLite.NET you can develop and deploy database applications to computers, or handheld, mobile, or embedded devices.
- **UltraLiteJ** This book describes UltraLiteJ. With UltraLiteJ, you can develop and deploy database applications in environments that support Java. UltraLiteJ supports BlackBerry smartphones and Java SE environments. UltraLiteJ is based on the iAnywhere UltraLite database product.

- **Error Messages** This book provides a complete listing of SQL Anywhere error messages together with diagnostic information.

Documentation conventions

This section lists the conventions used in this documentation.

Operating systems

SQL Anywhere runs on a variety of platforms. In most cases, the software behaves the same on all platforms, but there are variations or limitations. These are commonly based on the underlying operating system (Windows, Unix), and seldom on the particular variant (AIX, Windows Mobile) or version.

To simplify references to operating systems, the documentation groups the supported operating systems as follows:

- **Windows** The Microsoft Windows family includes Windows Vista and Windows XP, used primarily on server, desktop, and laptop computers, and Windows Mobile used on mobile devices.

  Unless otherwise specified, when the documentation refers to Windows, it refers to all Windows-based platforms, including Windows Mobile.

- **Unix** Unless otherwise specified, when the documentation refers to Unix, it refers to all Unix-based platforms, including Linux and Mac OS X.

Directory and file names

In most cases, references to directory and file names are similar on all supported platforms, with simple transformations between the various forms. In these cases, Windows conventions are used. Where the details are more complex, the documentation shows all relevant forms.

These are the conventions used to simplify the documentation of directory and file names:

- **Uppercase and lowercase directory names** On Windows and Unix, directory and file names may contain uppercase and lowercase letters. When directories and files are created, the file system preserves letter case.

  On Windows, references to directories and files are *not* case sensitive. Mixed case directory and file names are common, but it is common to refer to them using all lowercase letters. The SQL Anywhere installation contains directories such as *Bin32* and *Documentation*.

  On Unix, references to directories and files *are* case sensitive. Mixed case directory and file names are not common. Most use all lowercase letters. The SQL Anywhere installation contains directories such as *bin32* and *documentation*.

  The documentation uses the Windows forms of directory names. In most cases, you can convert a mixed case directory name to lowercase for the equivalent directory name on Unix.

- **Slashes separating directory and file names** The documentation uses backslashes as the directory separator. For example, the PDF form of the documentation is found in *install-dir\Documentation\en\PDF* (Windows form).

On Unix, replace the backslash with the forward slash. The PDF documentation is found in *install-dir/documentation/en/pdf*.

- **Executable files** The documentation shows executable file names using Windows conventions, with a suffix such as *.exe* or *.bat*. On Unix, executable file names have no suffix.

  For example, on Windows, the network database server is *dbsrv11.exe*. On Unix, it is *dbsrv11*.

- ***install-dir*** During the installation process, you choose where to install SQL Anywhere. The environment variable SQLANY11 is created and refers to this location. The documentation refers to this location as *install-dir*.

  For example, the documentation may refer to the file *install-dir*\readme.txt. On Windows, this is equivalent to *%SQLANY11%\readme.txt*. On Unix, this is equivalent to *$SQLANY11/readme.txt* or *${SQLANY11}/readme.txt*.

  For more information about the default location of *install-dir*, see "SQLANY11 environment variable" [*SQL Anywhere Server - Database Administration*].

- ***samples-dir*** During the installation process, you choose where to install the samples included with SQL Anywhere. The environment variable SQLANYSAMP11 is created and refers to this location. The documentation refers to this location as *samples-dir*.

  To open a Windows Explorer window in *samples-dir*, from the **Start** menu, choose **Programs** » **SQL Anywhere 11** » **Sample Applications And Projects**.

  For more information about the default location of *samples-dir*, see "SQLANYSAMP11 environment variable" [*SQL Anywhere Server - Database Administration*].

Command prompts and command shell syntax

Most operating systems provide one or more methods of entering commands and parameters using a command shell or command prompt. Windows command prompts include Command Prompt (DOS prompt) and 4NT. Unix command shells include Korn shell and bash. Each shell has features that extend its capabilities beyond simple commands. These features are driven by special characters. The special characters and features vary from one shell to another. Incorrect use of these special characters often results in syntax errors or unexpected behavior.

The documentation provides command line examples in a generic form. If these examples contain characters that the shell considers special, the command may require modification for the specific shell. The modifications are beyond the scope of this documentation, but generally, use quotes around the parameters containing those characters or use an escape character before the special characters.

These are some examples of command line syntax that may vary between platforms:

- **Parentheses and curly braces** Some command line options require a parameter that accepts detailed value specifications in a list. The list is usually enclosed with parentheses or curly braces. The documentation uses parentheses. For example:

  ```
  -x tcpip(host=127.0.0.1)
  ```

  Where parentheses cause syntax problems, substitute curly braces:

  ```
  -x tcpip{host=127.0.0.1}
  ```

If both forms result in syntax problems, the entire parameter should be enclosed in quotes as required by the shell:

```
-x "tcpip(host=127.0.0.1)"
```

- **Quotes** If you must specify quotes in a parameter value, the quotes may conflict with the traditional use of quotes to enclose the parameter. For example, to specify an encryption key whose value contains double-quotes, you might have to enclose the key in quotes and then escape the embedded quote:

  ```
  -ek "my \"secret\" key"
  ```

  In many shells, the value of the key would be `my "secret" key`.

- **Environment variables** The documentation refers to setting environment variables. In Windows shells, environment variables are specified using the syntax *%ENVVAR%*. In Unix shells, environment variables are specified using the syntax *$ENVVAR* or *${ENVVAR}*.

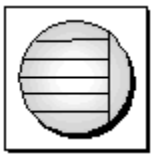
Graphic icons

The following icons are used in this documentation.

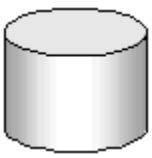
- A client application.

- A database server, such as Sybase SQL Anywhere.

- A database. In some high-level diagrams, the icon may be used to represent both the database and the database server that manages it.

- Replication or synchronization middleware. These assist in sharing data among databases. Examples are the MobiLink server and the SQL Remote Message Agent.

- A programming interface.

Contacting the documentation team

We would like to receive your opinions, suggestions, and feedback on this Help.

To submit your comments and suggestions, send an email to the SQL Anywhere documentation team at iasdoc@sybase.com. Although we do not reply to emails, your feedback helps us to improve our documentation, so your input is welcome.

DocCommentXchange

You can also leave comments directly on help topics using DocCommentXchange. DocCommentXchange (DCX) is a community for accessing and discussing SQL Anywhere documentation. Use DocCommentXchange to:

- View documentation
- Check for clarifications users have made to sections of documentation
- Provide suggestions and corrections to improve documentation for all users in future releases

Visit http://dcx.sybase.com.

Finding out more and requesting technical support

Additional information and resources are available at the Sybase iAnywhere Developer Community at http://www.sybase.com/developer/library/sql-anywhere-techcorner.

If you have questions or need help, you can post messages to the Sybase iAnywhere newsgroups listed below.

When you write to one of these newsgroups, always provide details about your problem, including the build number of your version of SQL Anywhere. You can find this information by running the following command: **dbeng11 -v**.

The newsgroups are located on the *forums.sybase.com* news server.

The newsgroups include the following:

- sybase.public.sqlanywhere.general
- sybase.public.sqlanywhere.linux
- sybase.public.sqlanywhere.mobilink
- sybase.public.sqlanywhere.product_futures_discussion
- sybase.public.sqlanywhere.replication
- sybase.public.sqlanywhere.ultralite
- ianywhere.public.sqlanywhere.qanywhere

For web development issues, see http://groups.google.com/group/sql-anywhere-web-development.

Newsgroup disclaimer

iAnywhere Solutions has no obligation to provide solutions, information, or ideas on its newsgroups, nor is iAnywhere Solutions obliged to provide anything other than a systems operator to monitor the service and ensure its operation and availability.

iAnywhere Technical Advisors, and other staff, assist on the newsgroup service when they have time. They offer their help on a volunteer basis and may not be available regularly to provide solutions and information. Their ability to help is based on their workload.

SQL Anywhere 11 Overview

This section introduces SQL Anywhere 11 and its data management and data exchange technologies.

Introducing SQL Anywhere

Contents

SQL Anywhere 11 overview

SQL Anywhere is a comprehensive package that provides technologies for data management and enterprise data exchange, enabling the rapid development of database-powered applications for server, desktop, mobile, and remote office environments.

SQL Anywhere offers:

- **Data management technologies** SQL Anywhere provides enterprise-caliber databases that are designed to handle the challenges of operating in many different frontline environments—from a high performance database server deployed with an independent software vendor application, to a mobile database that can be deployed to tens of thousands of handheld devices within the enterprise.

  See "Relational database systems" on page 6.

- **Data exchange technologies** SQL Anywhere offers several data exchange technologies to handle the complexities of exchanging data across unreliable wired and wireless networks to back-end databases, application servers, and messaging systems. In addition, SQL Anywhere mobile messaging and synchronization technologies guarantee secure message delivery for distributed and mobile computing.

  See "Data exchange technologies" on page 6.

- **Design and management tools** SQL Anywhere includes a suite of tools to improve the design and development of database-driven applications, and to simplify the management of databases and data exchange environments.

  See: "Design and management tools" on page 7.

SQL Anywhere in frontline environments

SQL Anywhere technologies are used in many different ways by over 10000 customers. Four common uses of SQL Anywhere are:

- **Client-server applications** Whether it is 5, 50, 500 users or more, SQL Anywhere is a powerful database solution for server applications, providing high performance out of the box, with low maintenance and cost.

  SQL Anywhere easily scales to support hundreds of active users, hundreds of gigabytes of data, and hundreds of millions of rows. Yet many ease-of-use and administration features ensure that costs stay down as performance scales up.

  This deployment model works best when the majority of users are connected to the network.

  See "Client/server applications and multi-user databases" on page 30.

- **Desktop applications** SQL Anywhere delivers enterprise-caliber features, but without the bulky characteristics of an enterprise database. Its robust reliability and performance, along with highly efficient usage of memory and system resources, ensure that the database can be hidden from laptop and desktop users.

Organizations embed SQL Anywhere databases in their applications because SQL Anywhere databases are built for use in widely-deployed, minimum administration environments, and require minimal memory and disk space.

See "Desktop applications and embedded databases" on page 30.

- **Remote office applications** SQL Anywhere data exchange architectures address the challenges of managing and sending data within and between offices and workers that are geographically distributed.

  Companies choose SQL Anywhere database and data exchange technologies to provide remote workers with the data they need to run their operations effectively, while providing the central office with the critical information that gives the pulse of the business.

  See "Consolidated and remote databases" on page 42.

- **Mobile and wireless applications** Recognized as the industry's leading mobile database, SQL Anywhere gives mobile workers the ability to have access to their data and corporate applications. Regardless of connection or application type, SQL Anywhere data exchange technologies ensure that mobile workers stay productive with the information they need, when they need it. Workers can access information and queue up transactions offline, reducing communications costs while increasing application and battery performance.

  Companies depend on SQL Anywhere for reliable management of data and mobile applications running on laptops, handheld devices, and smartphones.

  See:

  - "Introducing UltraLite" [*UltraLite - Database Management and Reference*]
  - "SQL Anywhere for Windows Mobile" [*SQL Anywhere Server - Database Administration*]

SQL Anywhere 11 components

The following sections describe the relational database systems, data exchange technologies, and design and management tools included in SQL Anywhere.

Relational database systems

SQL Anywhere offers two databases: SQL Anywhere Server and UltraLite.

SQL Anywhere Server

SQL Anywhere Server provides enterprise-caliber functionality including full transaction processing, referential integrity, materialized views, snapshot isolation, high availability via database mirroring and server clustering, SQL and Java stored procedures, triggers, row-level locking, automatic event scheduling, automatic backup and recovery, and much more. SQL Anywhere Server can easily scale to hundreds of concurrent users and hundreds of gigabytes of data. Yet its small footprint and its many features that automate administration make it an ideal database to embed into server and desktop applications that are widely deployed in customer and remote sites.

UltraLite

For environments that demand smaller data-driven applications, the UltraLite database is ideal. UltraLite is a full relational database management system designed specifically to minimize memory and system requirements for deployment to handhelds and other mobile devices, including the BlackBerry. It provides full transaction-processing, a choice of development models, and a synchronization client built directly into the database for exchanging data with other databases.

UltraLiteJ is a Java implementation of a subset of UltraLite, which is designed for BlackBerry smartphones and for Java ME. See "UltraLiteJ overview" [*UltraLiteJ*].

Data exchange technologies

SQL Anywhere offers a wide range of options for exchanging data with existing enterprise systems and mobile devices, including such tools as:

- **MobiLink—synchronization** MobiLink offers session-based, bi-directional synchronization. It is ideal for exchanging data between a central database and many remote UltraLite or SQL Anywhere databases, or between a central, non-relational data source and many remote UltraLite or SQL Anywhere databases.

  During a MobiLink synchronization, the remote database uploads changes that were made to it since the previous synchronization with the MobiLink server. On receiving this data, the MobiLink server updates the central database and then downloads changes from the central database to the remote database. It also ensures the transactional integrity of the databases in the event a connection between them is lost, and provides mechanisms for the resolution of data change conflicts.

MobiLink file transfer functionality lets you transfer files to remote applications on the same connection you use to synchronize data, which is useful when populating new remote databases or upgrading software.

In addition, MobiLink provides direct row handling for synchronizing remote data with any central data source. The data sources to which you can synchronize can include an application, web server, web service, application server, text file, spreadsheet, non-relational database, or an RDBMS that is not supported as a consolidated database.

- **QAnywhere—mobile enterprise messaging and mobile web services** QAnywhere offers a comprehensive mobile application-to-application messaging solution that provides secure, assured message delivery for distributed and mobile users. By extending the MobiLink server, QAnywhere reliably sends and receives messages between mobile applications and enterprise systems.

  QAnywhere offers:

  - Secure, store-and-forward messaging for smart-client applications
  - Reliable communication that is tolerant of network faults
  - Network-independent communication
  - Rules-based and guaranteed message delivery between occasionally-connected devices and message-based enterprise systems
  - Easy integration with Java Message Services—expanding your options for integrating enterprise systems with mobile database applications
  - Mobile web services

- **SQL Remote—replication** SQL Remote is a data-replication technology designed for two-way synchronization between a consolidated database and large numbers of remote databases, typically including many mobile databases.

  SQL Remote uses a store-and-forward architecture to synchronize data using a file or message transfer mechanism such as FTP or email.

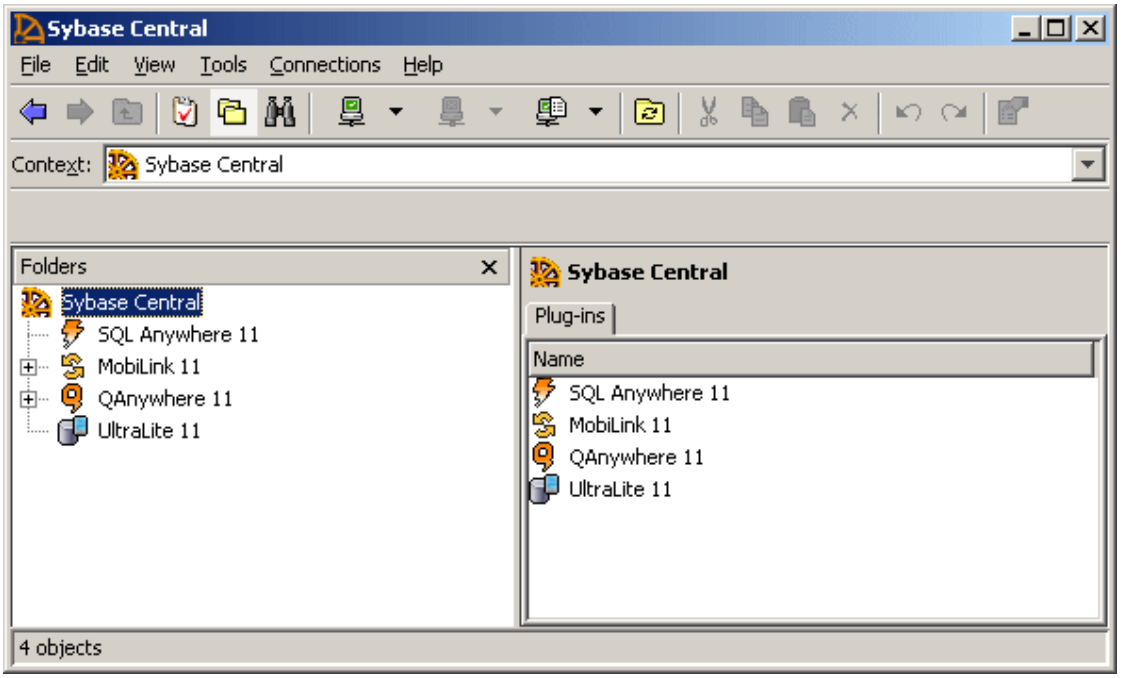
  SQL Remote preserves transactional integrity, making it ideal for many business applications, particularly those that operate in environments where connections are unreliable. Furthermore, memory and disk space requirements are minimal for all components of the replication system.

  See "SQL Remote introduction" [*SQL Remote*].

Design and management tools

The following describes the design and management tools included with SQL Anywhere.

- **Sybase Central—centralized control and administration** Sybase Central is an integrated database administration and development tool that provides access to database settings, properties, and utilities in a graphical user interface. Via plug-ins, Sybase Central can be used to manage SQL Anywhere Server, UltraLite, MobiLink, QAnywhere, and other Sybase products. For example, the MobiLink plug-in includes the **Create Synchronization Model Wizard** and **Model** mode to assist with developing and customizing MobiLink applications.



In addition to helping with routine tasks, Sybase Central also provides performance statistics, procedure profiling, and stored procedure debugging, and the management of events and schedules, web services, and connection profiles. Sybase Central helps administer any tasks that are performed by sending SQL statements to the database server, or performed by SQL Anywhere utilities. See "Using Sybase Central" [*SQL Anywhere Server - Database Administration*].

A variety of Sybase Central tools is available to help you analyze and monitor the current performance of your SQL Anywhere database. These tools include procedure profiling, graphical plans, query execution, the performance monitor, request logging, and timing utilities. In addition, Sybase Central offers:

- **Application profiling using the Application Profiling Wizard** Use the **Application Profiling Wizard** in Sybase Central to automatically:
  - Profile stored procedures, functions, triggers, and events
  - Receive recommendations to help improve the performance of your database application
  - Capture database activity while your application is running

  See "Application profiling" [*SQL Anywhere Server - SQL Usage*].
- **Advanced application profiling in Application Profiling mode** Improve overall performance by using the **Database Tracing Wizard** and **Application Profiling** mode in Sybase Central to:
  - Adjust cache size and indexes based on database performance counters
  - Identify when deadlocks occur
  - Look at locking activity

- Examine execution plans
- Trace each statement in an application for diagnosing and troubleshooting

See "Advanced application profiling using diagnostic tracing" [*SQL Anywhere Server - SQL Usage*].

- **Index selection and optimization using Index Consultant** The Index Consultant analyzes workloads and provides recommendations on how to select indexes to optimize performance. The Index Consultant can be run from either Sybase Central or Interactive SQL. See "Index Consultant" [*SQL Anywhere Server - SQL Usage*].

- **Interactive SQL—SQL query editor** Interactive SQL is a database utility designed to execute SQL statements and display database data. The built-in query editor and other tools, such as the graphical plan display, help you to analyze, troubleshoot, and optimize queries. See "Using Interactive SQL" [*SQL Anywhere Server - Database Administration*].
- **SQL Anywhere Monitor** The Monitor is a browser-based administration tool that provides you with information about the health and availability of SQL Anywhere databases and MobiLink servers. The Monitor provides constant data collection, email alert notifications, a browser-based interface, and the ability to monitor multiple databases and MobiLink servers. See "SQL Anywhere Monitor" [*SQL Anywhere Server - Database Administration*] and "SQL Anywhere Monitor for MobiLink" [*MobiLink - Server Administration*].
- **MobiLink Monitor—synchronization monitoring** The MobiLink Monitor is a graphical administration tool that provides details about the performance of MobiLink synchronizations. The MobiLink Monitor collects details and statistical summaries about all synchronizations that occur, including start and end times, data volume uploaded and downloaded, successful completions, conflicts, and more. See "MobiLink Monitor" [*MobiLink - Server Administration*].
- **Database utilities** SQL Anywhere includes various utilities for performing administration tasks such as backing up a database. Utilities are useful for including in batch files for repeated use. See "Database administration utilities" [*SQL Anywhere Server - Database Administration*].
- **InfoMaker** InfoMaker is a personal data assistant that lets you work with data in many ways. With InfoMaker you can query your SQL Anywhere databases and create sophisticated and effective custom reports. See "About InfoMaker" on page 72.
- **PowerDesigner Physical Data Model** The Physical Data Model component of Sybase PowerDesigner lets you design, generate, document, and maintain databases. See "About PowerDesigner Physical Data Model" on page 72.
- **DataWindow.NET** With DataWindow.NET you can rapidly build and deploy enterprise-level, SQL Anywhere applications that incorporate complex business rules, and deliver sophisticated data presentation. See "About DataWindow .NET" on page 72.

Editions and licensing

SQL Anywhere offers various **editions** that include certain separately licensed components and that can restrict the number of CPUs used by the database server. For more information about editions, visit http://www.sybase.com/products/databasemanagement/sqlanywhere/editions.

For more information on separately licensed components, see "Separately licensed components" on page 10.

Licensing and CPUs

With per-seat licensing, the network database server uses all CPUs available on the computer unless the database server is limited by the -gt option or by the edition you are running. With CPU-based licensing, the network database server uses up to the number of CPUs you are licensed for unless the database server is further limited by the -gt option or by the SQL Anywhere edition you are running.

The personal database server is limited to one CPU.

See also

- "-gt server option" [*SQL Anywhere Server - Database Administration*]

Separately licensed components

The following components are licensed separately and must be ordered from Sybase iAnywhere before you can install them. To order a separately licensed component, call Sybase iAnywhere at (800) 801-2069, or visit http://www.sybase.com/detail?id=1015780.

SQL Anywhere also offers various editions that include certain separately licensed components. See "Editions and licensing" on page 9.

SQL Anywhere security option

With SQL Anywhere you can strongly encrypt database files, and synchronization and client-server communication transport layers.

SQL Anywhere offers the following strong encryption algorithms:

	Included with a separately licensed security option[1]	Included with SQL Anywhere[2]
Database encryption	FIPS-approved AES	AES
Transport layer security	FIPS-approved RSA	RSA
Transport layer security	ECC	

[1] The software for strong encryption using ECC or FIPS-certified technology must be ordered separately.

[2] AES and RSA strong encryption are included with SQL Anywhere and do not require a separate license, but these libraries are not FIPS-certified.

The security option provides Certicom DLLs that implement the encryption algorithms, and additional DLLs that provide an interface between SQL Anywhere software and the Certicom libraries.

The SQL Anywhere security option includes the following:

- For Windows operating systems, Certicom Security Builder GSE.

For more information, see number 542 at http://csrc.nist.gov/cryptval/140-1/140val-all.htm.

- For Windows Mobile operating systems, Certicom Security Builder GSE.

  For more information, see number 316 at http://csrc.nist.gov/cryptval/140-1/140val-all.htm.

For more information about encryption, see "Keeping your data secure" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere CAC Authentication option

For UltraLite databases, the license for CAC Authentication must be ordered separately. In addition, this option requires the SQL Anywhere security option.

SQL Anywhere high availability option

For SQL Anywhere databases, the license for using the Veritas Cluster Server agents or for using database mirroring for failover must be ordered separately.

For information about platform support, see SQL Anywhere Supported Platforms and Engineering Support Status.

For more information about database mirroring, see "Introduction to database mirroring" [*SQL Anywhere Server - Database Administration*].

For more information about the SQL Anywhere Veritas Cluster Server agents, see "Using the SQL Anywhere Veritas Cluster Server agents" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere in-memory mode option

For SQL Anywhere databases, the license for using the in-memory mode must be ordered separately.

For information about platform support, see SQL Anywhere Supported Platforms and Engineering Support Status.

For more information about in-memory mode, see "-im server option" [*SQL Anywhere Server - Database Administration*].

MobiLink high availability option

The MobiLink high availability option allows you to group multiple MobiLink servers into server farms of identical servers using the shared state mode. When MobiLink runs in shared state mode it can block the same remote database from synchronizing simultaneously with multiple servers, thereby ensuring data integrity. MobiLink shared state support also enables load balancing and fail over for server initiated sync.

The MobiLink high availability option allows you to run MobiLink servers in shared state mode using the -ss option. See "-ss option" [*MobiLink - Server Administration*].

For more information about running MobiLink in a server farm, see "Running the MobiLink server in a server farm" [*MobiLink - Server Administration*].

Log Transfer Manager (LTM)

The LTM, which is the SQL Anywhere Replication Agent for Sybase Replication Server, is required for any SQL Anywhere database that participates in a Sybase Replication Server installation as a primary site. The license that is required for the LTM must be ordered separately.

For information about platform support, see SQL Anywhere Supported Platforms and Engineering Support Status.

Hallmarks of SQL Anywhere 11

The following is a list of SQL Anywhere hallmarks that you can take advantage of:

- **Embeddability** SQL Anywhere can be easily embedded inside other applications. It combines high performance with very small memory footprint. SQL Anywhere contains a range of features to enable self-management and maintenance in frontline environments, including features that enable optimization of computer resources, self-tuning for improved performance, and simplification of remote installation and support.
- **Interoperability** SQL Anywhere is available on many platforms, including Windows, Windows Mobile, Linux, Sun Solaris, HP-UX, IBM AIX, and Macintosh. Unique to SQL Anywhere, its database files can be copied between platforms. In addition, SQL Anywhere provides support for BlackBerry, Embedded Linux, Windows Mobile 6, and Java ME smartphones using its UltraLite database technology for small devices. SQL Anywhere includes support for many common database interfaces, including ODBC, JDBC, ADO.NET, PHP, and Perl. This means that many popular application development tools can be used, including: Microsoft Visual Studio, PowerBuilder, Eclipse, and various web tools. Stored procedures can be written in C/C++, Java, .NET, or Perl.
- **Performance out of the box** SQL Anywhere is designed to deliver outstanding performance, without ongoing tuning and administration. Features such as dynamic cache sizing, auto generation of statistics, a sophisticated query optimizer, parallel query processing, and materialized views mean that SQL Anywhere is ideal for environments that demand high performance but which have no on-site database administrator. By offering, On-Line Analytical Processing (OLAP) SQL Anywhere offers the ability to perform complex data analysis within a single SQL statement, increasing the value of the results, while improving performance by decreasing the amount of querying on the database.
- **Web operation** With a built-in HTTP server and web service support, XML features, full text search, and a PHP interface, SQL Anywhere is an ideal database for use in a web-based environment behind a web server.
- **Mobility** SQL Anywhere provides enterprise-caliber databases that operate on frontline systems and devices whether connectivity with enterprise systems is available or not. Its synchronization technologies ensure data can be exchanged efficiently over wireless and wired networks with back-end databases, application servers, and messaging systems.
- **Security** SQL Anywhere provides full end-to-end security with 128-bit strong encryption of database tables, files, and communications streams between the application and the database, and the MobiLink synchronization stream. SQL Anywhere can audit data access, offers built-in user authentication, and can integrate with third-party authentication systems. SQL Anywhere also offers FIPS-certified encryption via a separately licensed security option. See “SQL Anywhere security option” on page 10.

Supported platforms

The SQL Anywhere Supported Platforms and Engineering Support Status page lists the operating system platforms supported by SQL Anywhere, and indicating the engineering support status of previous versions of SQL Anywhere.

The web site reflects the support in the latest build of the software, so the information may not completely match your software. http://www.sybase.com/detail?id=1002288.

From this page, you can link to the following pages for more information about each SQL Anywhere component by platform.

- **SQL Anywhere Components by Platforms** http://www.sybase.com/detail?id=1061806
- **http://www.sybase.com/detail?id=1035824** http://www.sybase.com/detail?id=1035824
- **SQL Anywhere Supported Kerberos Clients** http://www.sybase.com/detail?id=1061807
- **SQL Anywhere Supported Listener Platforms** http://www.sybase.com/detail?id=1061808
- **MobiLink Redirector Supported Web Servers for SQL Anywhere** http://www.sybase.com/detail?id=1061837

For information about software updates, see "Checking for software updates" [*SQL Anywhere Server - Database Administration*].

Accessibility Enablement option

SQL Anywhere 11 includes an optionally-installable accessibility enablement module. This component provides the Sun Access Bridge module, which is loaded whenever you use Sybase Central or Interactive SQL. Third-party software such as screen readers use of this module to provide access to software features.

For information about platform support for the accessibility enablement option, see http://www.sybase.com/detail?id=1002288.

For more information about SQL Anywhere accessibility, see http://www.sybase.com/accessibility.

Overview of data management technologies

Contents

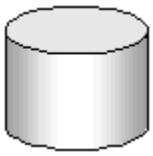
The parts of a database system

A **relational database management system** (RDBMS) is a system for storing and retrieving data, in which the data is organized into interrelated tables.

Relational database management systems contain the following pieces:

- a database
- a database server
- an application programming interface (API)
- a client application

Database Data is stored in a database. In diagrams in the documentation, a database is indicated by a cylinder:



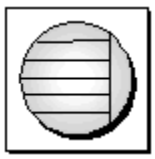
A SQL Anywhere database is a file, usually with an extension of *.db*. An UltraLite database is also a file, usually with an extension of *.udb*. SQL Anywhere includes a sample database for you to work with, and it is installed in the SQL Anywhere samples directory: *samples-dir\demo.db*.

For information about the default location of *samples-dir*, see "Samples directory" [*SQL Anywhere Server - Database Administration*].

Database server The database server manages the database. All access to the database occurs through the database server.

The database server allows access to databases from client applications, and processes commands in a secure and efficient manner. A database can have only one server managing it at a time. However, a SQL Anywhere database server can manage many databases at one time.

In the documentation, a database server is indicated as follows:



There are two versions of the SQL Anywhere database server: the **personal server** and the **network server**. Both servers offer the same query processing and other internal operations; the only difference is in the number and types of connections each server accepts.

The personal server only accepts a maximum of ten concurrent connections from applications or users running on the same computer. It is intended for single-user, same-computer use.

By contrast, the network server supports client/server communication over a network and is intended for multi-user, multi-computer use. The maximum number of connections is governed by your license agreement.

See "Two types of SQL Anywhere database servers" on page 25.

UltraLite runtime library In UltraLite, the database management systems that are typically found in a database server are implemented as an in-process runtime library. The runtime library and the application are part of the same process.

Programming interface Applications communicate with the database server using a programming interface such as ODBC, JDBC, OLE DB, ADO.NET, or embedded SQL.

For a complete list of supported programming interfaces in SQL Anywhere and UltraLite, see "Programming interfaces" on page 35.

Each programming interface provides a library of function calls for communicating with the database. For ODBC and JDBC, the library is commonly called a **driver**. The library is typically provided as a shared library on Unix operating systems or a dynamic link library (DLL) on Windows operating systems.

In diagrams in the documentation, a programming interface is indicated as follows:

Client application Client applications use one of the programming interfaces to communicate with the database server.

If you develop an application using a rapid application development (RAD) tool such as Sybase PowerBuilder, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications use one of the supported interfaces.

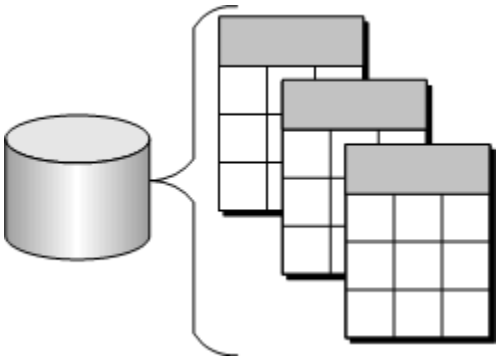
In diagrams in the documentation, a client application is indicated as follows:

Relational database concepts

The following sections provide a brief review of basic relational database concepts. They include definitions of tables, primary and foreign keys, and database objects.

Database tables

In a relational database, all data is held in **tables**, which are made up of **rows** and **columns**.



Each table has one or more columns, and each column is assigned a specific data type, such as an integer, a sequence of characters (for text), or a date. Each row in the table has a single value for each column.

For example, a table containing employee information can look like the following:

EmployeeID	Surname	GivenName	Phone
102	Huong	Zhang	1096
10693	Donaldson	Anne	7821

Characteristics of relational tables

The tables in a relational database have some important characteristics:

- There is no significance to the order of the columns or rows.
- Each row contains one and only one value for each column, or contains NULL, which indicates that there is no value for that column.
- All values for a given column have the same data type.

The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their equivalent in non-relational databases such as dBase and FoxPro. This document uses the informal terms.

Informal relational term	Formal relational term	Non-relational term
Table	Relation	File
Column	Attribute	Field
Row	Tuple	Record

What do you keep in each table?

Each table in the database should hold information about a specific kind of thing, such as employees, products, or customers.

By designing a database this way, you can set up a structure that eliminates redundancy and the possible inconsistencies caused by redundancy. For example, both the sales and accounts payable departments might enter and look up information about customers. In a relational database, the information about customers is stored only once, in a table that both departments can access.

See "Creating databases in SQL Anywhere" [*SQL Anywhere Server - SQL Usage*].

Relations between tables

You use primary keys and foreign keys to describe relationships between the information in different tables. **Primary keys** identify each row in a table uniquely, and **foreign keys** define the relationships between rows in different tables.

Primary keys and foreign keys let you use relational databases to hold information in an efficient manner, with minimum redundancy.

Primary keys

Each table in a relational database should have a **primary key**. The primary key is a column, or set of columns, that uniquely identifies each row. No two rows in a table can have the same primary key value.

Examples

In the SQL Anywhere sample database (*samples-dir\demo.db*), the Employees table stores personal information about employees. It has a primary key column named EmployeeID, which holds a unique ID number assigned to each employee. A single column holding an ID number is a common way to assign primary keys, and has advantages over names and other identifiers that may not always be unique.

A more complex primary key can be seen in the SalesOrderItems table of the SQL Anywhere sample database. The table holds information about individual items on orders from the company, and has the following columns:

- **ID** An order number, identifying the order the item is part of.
- **LineID** A line number, identifying each item on any order.

- **ProductID** A product ID, identifying the product being ordered.
- **Quantity** A quantity, displaying how many items were ordered.
- **ShipDate** A ship date, displaying when the order was shipped.

A particular sales order item is identified by the order it is part of and by a line number on the order. These two numbers are stored in the ID and LineID columns. Items can share a single ID value (corresponding to an order for more than one item) or they can share a LineID number (all first items on different orders have a LineID of 1). No two items share both values, and so the primary key is made up of these two columns.

Foreign keys

The information in one table is related to that in other tables by **foreign keys**.

Example

The SQL Anywhere sample database has one table holding employee information and one table holding department information. The Departments table has the following columns:

- **DepartmentID** An ID number for the department. This is the primary key for the table.
- **DepartmentName** The name of the department.
- **DepartmentHeadID** The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the Employees table. Instead, the Employees table contains a column holding a number that matches one of the DepartmentID values in the Departments table.

The DepartmentID column in the Employees table is called a foreign key to the Departments table. A foreign key references a particular row in the table containing the corresponding primary key.

In this example, the Employees table (which contains the foreign key in the relationship) is called the **foreign table** or **referencing table**. The Departments table (which contains the referenced primary key) is called the **primary table** or the **referenced table**.

Other database objects

There is more to a relational database than simply a set of related tables. You can also find the following objects in a relational database:

- **Indexes** Indexes allow quick look up of information. Conceptually, an index in a database is like an index in a book. In a book, the index relates each indexed term to the page or pages on which that word appears. In a database, the index relates each indexed column value to the physical location at which the row of data containing the indexed value is stored.

  Indexes are an important design element for high performance. You usually must create indexes explicitly, but indexes for primary, foreign keys, and for unique columns are created automatically. Once

created, the use of indexes is transparent to the user. See "Indexes" [*SQL Anywhere Server - SQL Usage*].

- **Text indexes** Text indexes store complete positional information for every instance of every term in every indexed column. When you perform a full text search, a text index is used to find matching rows. For this reason, queries that use text indexes can be faster than those that must scan all the values in the table. See "Text indexes" [*SQL Anywhere Server - SQL Usage*], and "Full text searching" [*SQL Anywhere Server - SQL Usage*].
- **Login policies** Login policies consist of a set of rules that are applied when you create a database connection for a user. See "Managing login policies overview" [*SQL Anywhere Server - Database Administration*].
- **Views** Views are temporary tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.

  The tables that actually hold the information are sometimes called **base tables** to distinguish them from views. A view is defined with a SQL query on base tables or other views.

  See "Working with views" [*SQL Anywhere Server - SQL Usage*].
- **Materialized views** SQL Anywhere also supports materialized views. A materialized view is a view whose result set has been computed and stored on disk, similar to a base table. Conceptually, a materialized view is both a view (it has a query specification) and a table (it has persistent materialized rows). So, many operations that you perform on tables can be performed on materialized views as well. For example, you can build indexes on, and unload from, materialized views.

  Materialized views are ideal for environments where the database is large, frequent queries result in repetitive aggregation and join operations on large amounts of data, and access to up-to-the-moment data is not a critical requirement. See "Working with materialized views" [*SQL Anywhere Server - SQL Usage*].
- **Stored procedures and triggers** These are routines held in the database that act on the information in the database.

  You can create and name your own stored procedures to execute specific database queries and to perform other database tasks. Stored procedures can accept parameters and return result sets. For example, you might create a stored procedure that returns the names of all customers who have spent more than the amount that you specify as a parameter in the call to the procedure.

  A trigger is a special stored procedure that automatically fires whenever a user updates, deletes, or inserts data, depending on how you define the trigger. You associate a trigger with a table or columns within a table. Triggers are useful for automatically maintaining business rules in a database.

  You can also install Java classes into the database. Java classes provide a powerful way of building logic into your database. See "Creating a Java class for use with SQL Anywhere" [*SQL Anywhere Server - Programming*].

  See "Using procedures, triggers, and batches" [*SQL Anywhere Server - SQL Usage*].
- **Users and groups** Each user of a database has a user ID and password. You can set permissions for each user so that confidential information is kept private and users are prevented from making unauthorized changes. Users can be assigned to groups to make the administration of permissions easier.

See “Managing user IDs, authorities, and permissions” [*SQL Anywhere Server - Database Administration*].

In addition to these common database objects, SQL Anywhere also provides advanced features:

- Events
- Domains
- Publications
- Web services
- Remote data access
- Maintenance plans

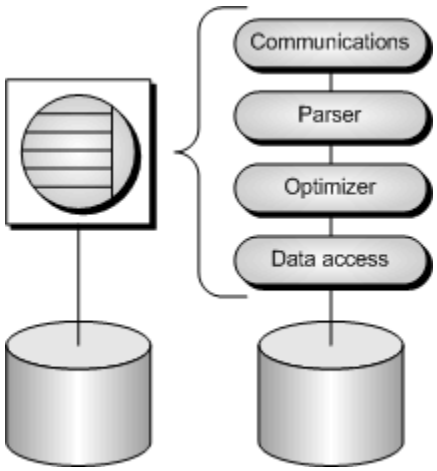
Inside SQL Anywhere

While you never need to deal with the internals of the database server, a glimpse behind the scenes can help you understand how the database server and the database interact.

Inside the database server

The SQL Anywhere database server has an internal structure that allows many requests to be handled efficiently.

- A communications layer handles the exchange of data with client applications. This layer receives requests from client applications, and returns results. The timing of these actions is governed by a negotiation between the client and the server to make sure that the network traffic is kept to a minimum, but that the data is made available as soon as possible on the client side.
- The parser checks each SQL statement sent to the database server, and transforms it into an internal form for processing.
- If the request is a query, an update, or delete statement, there can be many different ways of accessing the data, which may take significantly different times. The optimizer selects the best method of getting the required data quickly.
- The lowest level of the database server is concerned with reading and writing data from the disk, caching data in memory to avoid unnecessary disk access, and balancing the demands of different users.



Two types of SQL Anywhere database servers

There are two versions of the SQL Anywhere database server: the **personal server** and the **network server**.

The request-processing engine is identical in both versions, and they support exactly the same SQL language, and exactly the same database features. However, the personal server does not support communications across a network, more than ten concurrent connections, or the use of more than one computer. Applications

developed against a personal server work unchanged against a network server. Other differences are as follows:

- The **personal server** can only accept connections from applications or users running on the same computer. It is intended for single-user, same-computer use: for example, as an embedded database server. It is also useful for development work.

  The name of the personal server executable is as follows:

  - On Windows it is *dbeng11.exe*. (It is not provided on Windows Mobile.)
  - On Unix operating systems, it is *dbeng11*.
- By contrast, the **network server** supports client/server communication over a network and is intended for multi-user operation.

  The name of the network server executable is as follows:

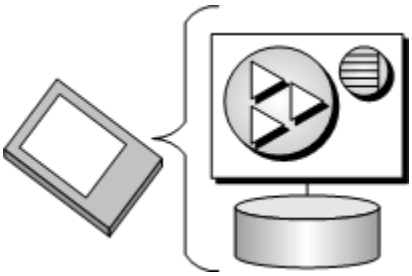
  - On Windows, including Windows Mobile, it is *dbsrv11.exe*. The network server is supplied for Windows Mobile so that the desktop applications can connect to databases in the mobile device.
  - On Unix operating systems, it is *dbsrv11*.

If you have received SQL Anywhere as part of another product, you may not have both versions of the database server. Similarly, not all components are available on all operating systems. For example, there is no personal server for Windows Mobile, only a network server.

For more information about running the personal and network database servers, see "Running the database server" [*SQL Anywhere Server - Database Administration*].

Inside UltraLite

If you want to provide a database application for a small device such as a handheld organizer, you may want to use UltraLite. In UltraLite, the functions performed by the server are typically placed in a runtime library. The runtime library is combined with the application to become part of the same process. So, there is a one-to-one relationship between the database and the application.



For deployments that require multiple applications to connect concurrently to one database on the same device, the library must exist as a separate process. In these cases, the UltraLite database engine is used.

Other features

- UltraLite has built-in MobiLink synchronization technology so that your application is linked into the information network.

For information about integrating UltraLite and MobiLink, see "UltraLite clients" [*UltraLite - Database Management and Reference*].

- UltraLite supports many operating systems, see "Introducing UltraLite" [*UltraLite - Database Management and Reference*].

Database files

The following sections provide an overview of the type of files, including database, transaction, and temporary files, that comprise a database. Dfferences between the implementation of these files in SQL Anywhere and UltraLite are also discussed.

SQL Anywhere database files

All the information in a SQL Anywhere database is usually stored in a single database file, which can be copied from one computer to another. It is possible have a database made up of several files, but this is generally only required for very large databases.

In addition to the database file, SQL Anywhere uses two other files when running a database: the transaction log and the temporary file.

- **The database file** Internally, the database file is composed of pages: fixed size areas of disk. The data access layer reads and writes data one page at a time. Many pages hold the data that is in the database tables, but other pages hold index information, information about the distribution of data within the database, and so on.
- **The transaction log** The transaction log is a separate file that contains a record of all the operations performed on the database. Normally, the transaction log has the same name as the database file, except that it ends with the suffix *.log* instead of *.db*. It has three important functions:
  - **Record operations on your data to enable recovery** You can recreate your database from a backup together with the transaction log if the database file is damaged.
  - **Improve performance** By writing information to the transaction log, the database server can safely process your statements without writing to the database file as frequently.
  - **Enable database replication** SQL Remote and MobiLink synchronization use the transaction log to synchronize changes to your other databases.
- **The temporary file** The temporary file is created when the database server starts, and is erased when the database server stops. As its name suggests, the temporary file is used while the database server is running to hold temporary information. The temporary file does not hold information that needs to be kept between sessions.

  The UltraLite temporary file is stored in the same directory as the database file.

  See "TMP, TEMPDIR, and TEMP environment variables" [*SQL Anywhere Server - Database Administration*].

Inside the UltraLite database

UltraLite databases contain the same features as described above with the following exceptions:

- UltraLite database files don't contain information about the distribution of data within the database.
- UltraLite keeps track of its transactions internally, not in a separate log file.
- The UltraLite temporary file is stored in the same directory as the database file.

See “UltraLite transaction and state management” [*UltraLite - Database Management and Reference*].

Choosing between SQL Anywhere and UltraLite databases

SQL Anywhere and UltraLite address data storage and data access needs from large enterprise database sources to small, mobile databases. When designing an application, you need to choose the database that is the right fit.

- If your target platform is Unix or Mac OS X, you must use a SQL Anywhere database.
- If your target platform is the Palm OS, BlackBerry, Embedded Linux, Windows Mobile 6, or Java ME smartphones, you must use an UltraLite database.
- If the target platform is Windows Vista, Windows XP, or Linux, both SQL Anywhere and UltraLite are available. SQL Anywhere is often preferred because provides a fuller feature set and its additional memory requirements are rarely an issue.
- If the target platform is Windows Mobile, such as on a Pocket PC or smartphone, you need to consider memory constraints, and possibly the tasks your application needs to perform. On Windows Mobile, SQL Anywhere requires approximately 6 MB of memory plus another 2 MB for the synchronization component, whereas UltraLite requires less than 1 MB and has synchronization built in. But, while UltraLite is considerably smaller, it does not offer the same support as SQL Anywhere for such things as complex queries, events, procedures, triggers, views, and so on.

Your overall solution may result in a mixture of SQL Anywhere and UltraLite databases, synchronized using MobiLink.

For more information about the differences between the core database solution (SQL Anywhere) and the UltraLite database solution, see "UltraLite feature comparison" [*UltraLite - Database Management and Reference*].

Database scenarios

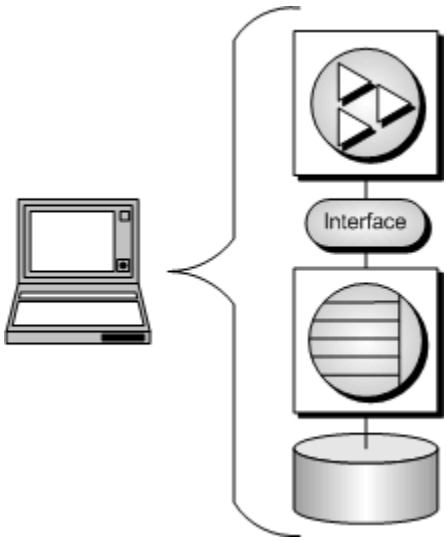
Database applications can connect to a database server located on the same computer as the application, or in the case of the network database server, on a different computer. In addition, with SQL Anywhere you can build distributed databases for remote office and mobile applications, with physically distinct databases on different computers sharing data.

Desktop applications and embedded databases

You can use SQL Anywhere to build a complete application and database on a single computer. In the simplest arrangement, this is a **standalone application** or **personal application**: it is self-contained with no connection to other databases. In this case, the database server and the database can be started by the client application, and it is common to refer to the database as an **embedded database**. As far as the end user is concerned, the database is a part of the application.

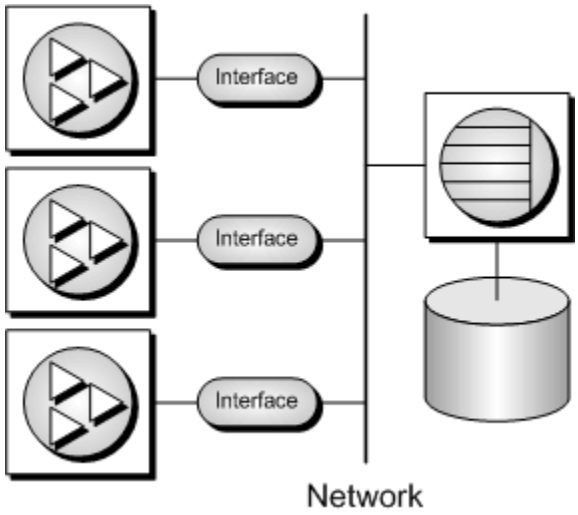
Many relational database management systems require experienced staff for administration. A characteristic of SQL Anywhere databases is the ability to run entirely without administration.

The SQL Anywhere personal database server is generally used for embedded applications. Embedded applications have the following architecture, with a client application connecting through a programming interface to a database server running on the same computer:



Client/server applications and multi-user databases

You can use SQL Anywhere to build an installation with many applications running on different computers, connected over a network to a single network database server running on a separate computer. This is a **client/server** or **multi-user database** environment, and has the following architecture. The interface library is located on each client computer.



In this case, the database server is the SQL Anywhere network database server, which supports network communications.

For a client application to work in a client/server environment, you need to specify additional connection parameters, minimally the server name and the communication protocol.

See also

- "Types of deployment" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere database connections" [*SQL Anywhere Server - Database Administration*]

Multi-tier computing architecture

In multi-tier computing, application logic is held in an application server, such as Sybase EAServer, WebLogic, or WebSphere, which sits between the database server and the client applications. In many situations, a single application server can access multiple databases in addition to non-relational data stores. In the Internet case, client applications are browser-based, and the application server is generally a web server extension. Many modern multi-tier applications use a service-oriented architecture (SOA) based on web services.

Sybase EAServer stores application logic in the form of components, and makes these components available to client applications. The components can be PowerBuilder components, Java beans, or COM components.

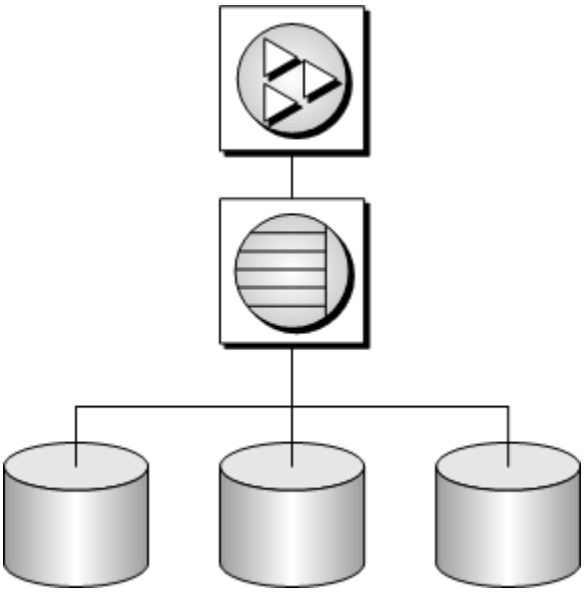
Application servers can also provide transaction logic to their client applications—guaranteeing that sets of operations are executed atomically across multiple databases. SQL Anywhere is well suited to multi-tier computing, and can participate in distributed transactions coordinated by Microsoft Distributed Transaction Coordinator. Both Sybase Enterprise Application Server and Microsoft Transaction Server use DTC to provide transaction services to their client applications.

The built-in support for web services makes SQL Anywhere a good choice for many multi-tier or SOA applications.

See “Three-tier computing and distributed transactions” [*SQL Anywhere Server - Programming*].

Running multiple databases on a single database server

Both the SQL Anywhere personal database server and the network database server can manage many databases simultaneously. Each connection from an application must be to a single database, but an application can use separate connections to different databases, or a set of applications can work on different databases, all through the same database server.



Databases can be started when the database server is started, by connecting to a database using the DatabaseFile connection parameter, or by using the START DATABASE statement.

See also

- "The SQL Anywhere database server" [*SQL Anywhere Server - Database Administration*]
- "DatabaseFile connection parameter [DBF]" [*SQL Anywhere Server - Database Administration*]
- "START DATABASE statement" [*SQL Anywhere Server - SQL Reference*]

Accessing data in other databases

You can access databases on multiple database servers, or even on the same server, using SQL Anywhere remote data access. The application is still connected to a single database, but by defining remote servers, you can use proxy tables that exist on the remote database as if they were in the database to which you are connected. See "Accessing remote data" [*SQL Anywhere Server - SQL Usage*].

ETL features

Extract, Transform, and Load (ETL) is the process by which large amounts of data is extracted from disparate data sources and consolidated into a single database. In the extraction phase, data is parsed and evaluated for suitability. During transformation, data is manipulated to achieve the format required for storage. Some common transformations include the elimination of unnecessary columns, calculation of computed values, and translation of values such as dates into a common format so that the data can be consolidated. The data is then loaded into the database at a frequency and scope consistent with the organization's needs.

SQL Anywhere offers several features in support of ETL. For example:

- **OPENSTRING operation** Use the OPENSTRING operation in the FROM clause to transform and load data from client- and server-side data sources. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].
- **openxml system procedure** Use the openxml system procedure to extract data from XML documents. See "openxml system procedure" [*SQL Anywhere Server - SQL Reference*].
- **MERGE statement** Use the MERGE statement to merge data from different source objects. See "MERGE statement" [*SQL Anywhere Server - SQL Reference*].
- **Proxy tables** Use proxy tables to access objects such as tables, views, and materialized views in a remote database. See "Working with proxy tables" [*SQL Anywhere Server - SQL Usage*].
- **System procedure calls in the FROM clause** You can use various system procedures in the FROM clause of a query to extract and transform data for loading. For a list of system procedures offered in SQL Anywhere, see "System procedures" [*SQL Anywhere Server - SQL Reference*].

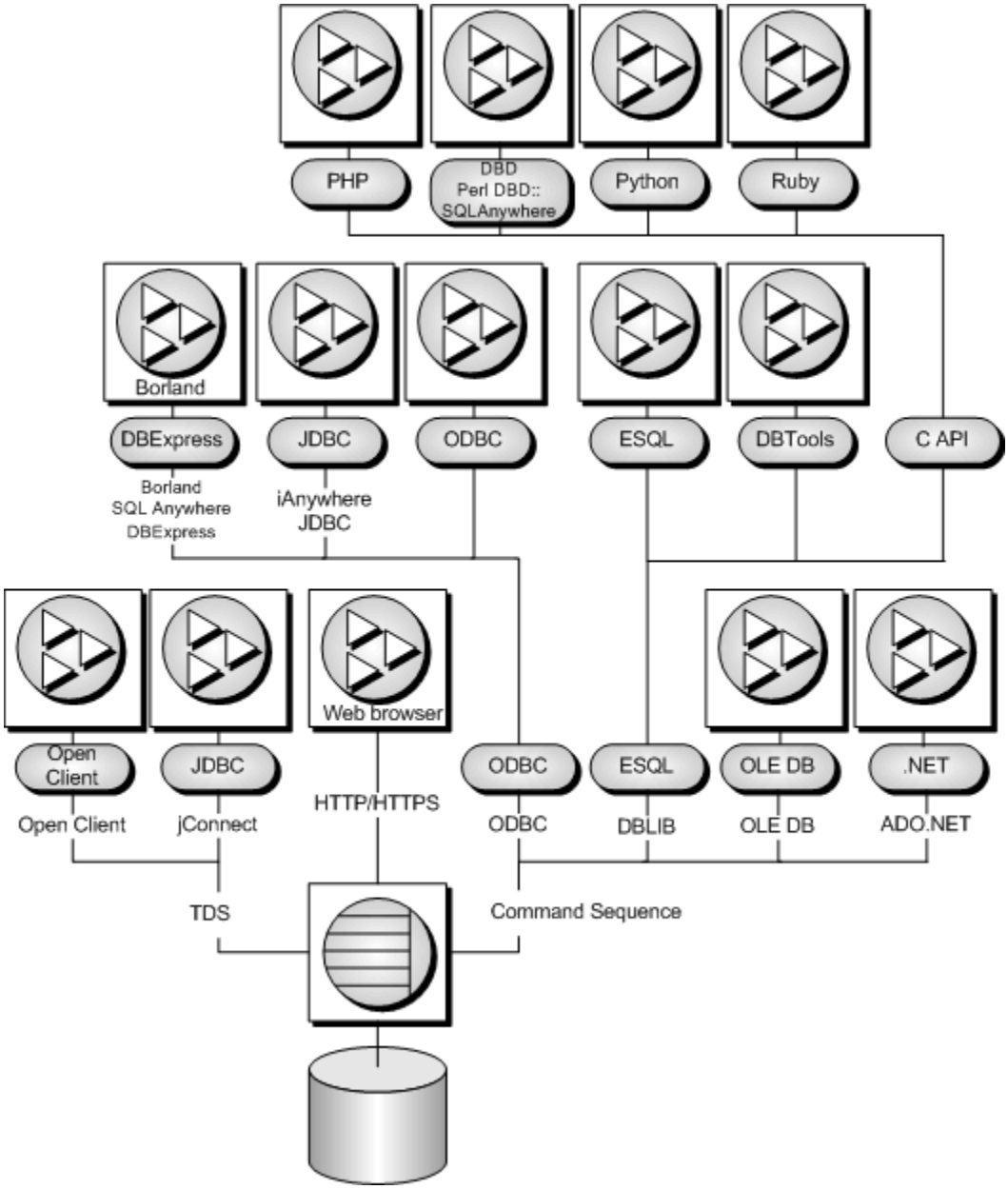
Programming interfaces

SQL Anywhere supports a wide set of data access programming interfaces to provide flexibility in the kinds of applications and application development environments you can use.

For an overview of database application architecture, see “Database scenarios” on page 30.

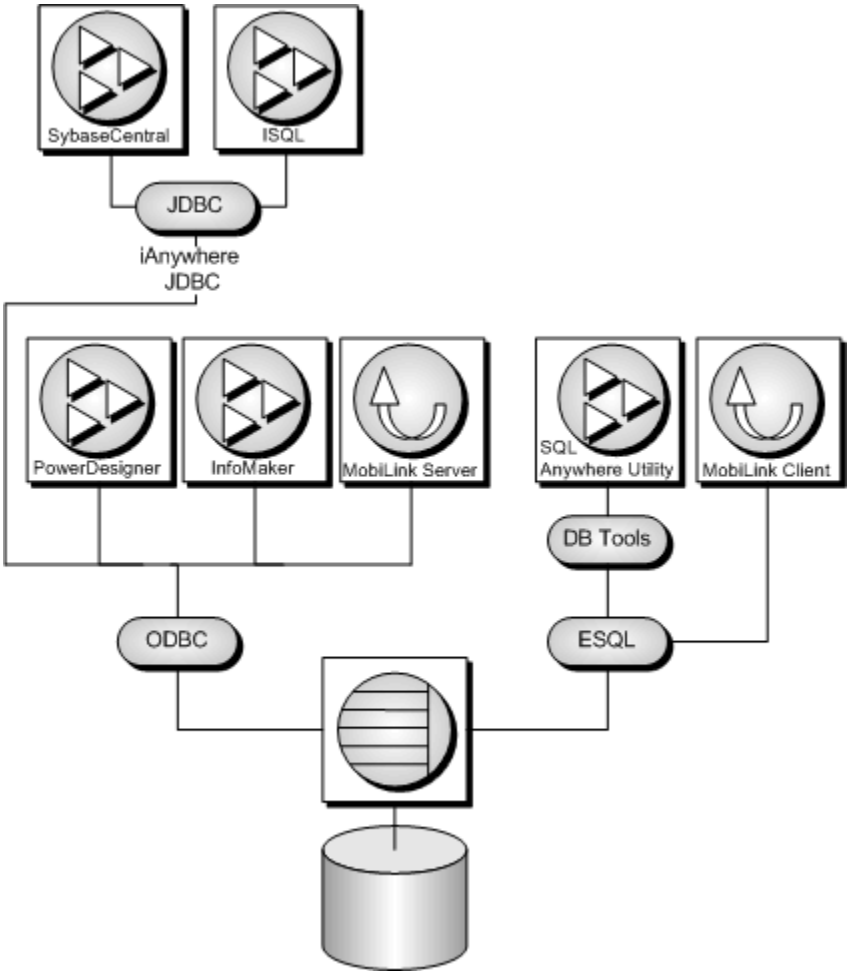
Supported programming interfaces and protocols

The following diagram displays the supported interfaces, and the interface libraries used. In most cases, the interface library has the same name as the interface.



SQL Anywhere applications

The applications supplied with SQL Anywhere use several of these interfaces:



SQL Anywhere programming interfaces

For specific details about the SQL Anywhere programming interfaces, see the list below:

- "SQL Anywhere ODBC support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere .NET support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere OLE DB and ADO support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere embedded SQL" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere JDBC support" [*SQL Anywhere Server - Programming*]
- "Sybase Open Client support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere Perl DBI support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere PHP support" [*SQL Anywhere Server - Programming*]

UltraLite programming interfaces

UltraLite also provides developers a choice of programming interfaces for straightforward access to data. The following is a list of the UltraLite programming interfaces:

- C/C++
- Embedded SQL using C/C++
- UltraLite.NET using C# or VB.NET
- M-Business Anywhere using JavaScript

For more information about UltraLite programming interfaces, see "Choosing an UltraLite programming interface" [*UltraLite - Database Management and Reference*].

Communications protocols

Each interface library communicates with the database server using a **communication protocol**. SQL Anywhere supports two communication protocols, **Command Sequence** and **Tabular Data Stream** (**TDS**). These protocols are internal, and for most purposes it does not matter which one you are using. Your choice of development environment will be governed by your available tools.

The major differences are visible when connecting to the database. Command Sequence applications and TDS applications use different methods to identify a database and database server, and so connection windows are different.

Command Sequence This protocol is used by SQL Anywhere, the iAnywhere JDBC driver, and the embedded SQL, ODBC, OLE DB, and ADO.NET APIs.

TDS This protocol is used by Sybase Adaptive Server Enterprise, Open Client applications, and Java applications that use the jConnect JDBC driver connect using TDS.

Overview of data exchange technologies

Contents

Comparing synchronization technologies

Data exchange technologies include synchronization, replication, messaging, and mobile web service technologies.

Data **synchronization** is the sharing of data among physically distinct databases. When an application modifies shared data at any one database, the changes are propagated to other databases in the synchronization system. Changes can be propagated by various means and through a variety of channels, allowing flexible application architecture while preserving data integrity.

SQL Anywhere offers two synchronization technologies:

- **MobiLink** is a session-based technology intended for the one- or two-way synchronization of data between a central, consolidated database and a large number of remote databases. It supports a variety of consolidated database servers, and provides an API for synchronizing with virtually any other data source. Administration and resource requirements at the remote sites are minimal, making MobiLink well suited to a variety of mobile applications. At the end of each synchronization session, the databases are consistent.
- **SQL Remote** is a message-based technology intended for the two-way replication of database transactions. It is designed for two-way replication involving a consolidated data server and large number of remote databases. Administration and resource requirements at the remote sites are minimal, making SQL Remote well suited to mobile databases.

The following table summarizes the characteristics of MobiLink and SQL Remote.

Synchronization technology	Number of databases	Connection	Frequency	Consolidated database types
MobiLink	Large	Occasional	Medium	Many options
SQL Remote	Large	Occasional	Low	SQL Anywhere

MobiLink characteristics

MobiLink is designed for synchronization systems with the following requirements:

- **Large numbers of remote databases** MobiLink is designed to support large numbers of remote databases. It can handle tens of thousands of simultaneous synchronizations.
- **Occasionally connected** MobiLink supports databases that are occasionally connected or indirectly connected to the network on which the server is running.
- **Consolidated databases supported** MobiLink supports virtually any type of data store as the central data source. The remote data stores must be SQL Anywhere or UltraLite databases. The schema of the remote sites can be different from that of the consolidated database because you control the synchronization process by writing scripts.

- **Flexible synchronization schedule** Applications can connect and synchronize at intervals of seconds, minutes, hours, or days.

SQL Remote characteristics

SQL Remote is designed for synchronization systems with the following requirements:

- **Large numbers of remote databases** SQL Remote is designed to support a large number of remote databases. It can support thousands of remote databases in a single installation because the messages for many remote sites can be prepared simultaneously.
- **Occasionally connected** SQL Remote supports databases that are occasionally connected or indirectly connected to the network.
- **Low to high latency** High latency means a long lag time between data being entered at one database and being replicated to each database in the system. With SQL Remote, replication messages can be sent at periods of seconds, minutes, hours, or days.
- **Low to moderate volume** As replication messages are delivered occasionally, a high transaction volume at each remote site can lead to a very large volume of messages. SQL Remote is best suited to systems with a relatively low volume of replicated data per remote database. However, at the consolidated site, SQL Remote can prepare messages efficiently by preparing messages for multiple sites simultaneously.
- **Homogeneous databases** SQL Remote supports SQL Anywhere databases. Each database in the system must have a similar schema.

Benefits of data synchronization

Data availability

One of the key benefits of a data synchronization system is that data is made available locally, rather than through potentially expensive, less reliable, and slow connections to a single central database. Data is accessible locally even in the absence of any connection to a central database, so you are not cut off from data in the event of a failure of a network connection.

Response time

Synchronization improves response times for data requests for two reasons. Retrieval rates are faster because requests are processed on a local server, without accessing a wide area network. Also, local processing offloads work from a central database server so that competition for processor time is decreased.

Challenges for synchronization technologies

Any synchronization technology must address several challenges that arise as a result of the increased flexibility permitted by synchronization.

Transactional integrity

One of the challenges of any synchronization system is to ensure that each database always retains transactional integrity.

SQL Remote replicates portions of the transaction log in such a way that transactions are maintained during synchronization: either a whole transaction is replicated, or none of it is replicated. This ensures transactional integrity at each database in the system.

In MobiLink, you can also choose to replicate each transaction, but by default MobiLink coalesces multiple transactions on the remote database and applies them in a single transaction. This generally results in more efficient uploads. In both cases, MobiLink maintains transactional integrity.

Data consistency

Another challenge to synchronization systems is to maintain data consistency throughout the system. Synchronization systems maintain a **loose consistency** in the system as a whole: that is, all changes are replicated to each site over time in a consistent manner, but different sites may have different copies of data at any instant.

See also

- "Synchronization techniques" [*MobiLink - Server Administration*]

Consolidated and remote databases

Both MobiLink and SQL Remote provide data synchronization between a central database and a set of remote databases.

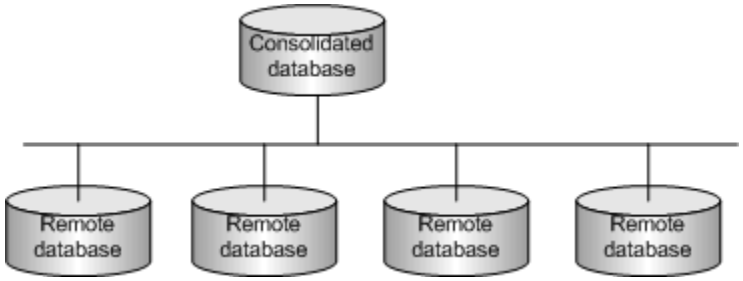
In MobiLink, the **consolidated database** is one of several supported RDBMSs. The consolidated database, which typically resides on a corporate server, tracks synchronization information and optionally contains the data to be replicated. Other central data may be stored in any other format, such as a non-relational database, web service, or text file.

MobiLink also provides direct row handling, which enables data synchronization to consolidated data sources other than relational databases including enterprise resource planning (ERP) systems or application servers.

In SQL Remote, all data that is to be synchronized is contained in a SQL Anywhere consolidated database.

A **remote database** can run either at the same site as the consolidated database or at a physically distant site such as a handheld device. The remote database can share all or some of the data in the consolidated database.

The following figure displays a schematic illustration of a small synchronization system.



Remote users

A typical synchronization system includes many remote databases. Each remote database contains a subset of the information in the central database. Each remote database is a physically separate database, usually on a separate computer or mobile device. All remote databases must stay consistent with the central database.

The entire synchronization system may be considered a single dispersed database, with the master copy of all shared data being kept at the central database.

Each remote site that synchronizes with the central database is considered to be a remote user of the central database. In the case that a remote site is a multi-user server, the entire site is considered to be a single remote user of the central database.

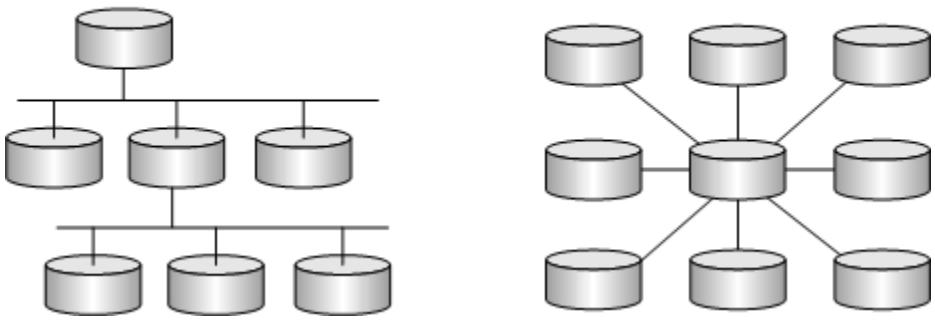
Hierarchical database configurations

For databases in a **hierarchical configuration**, every database has a single parent database, except the consolidated database, which has no parent.

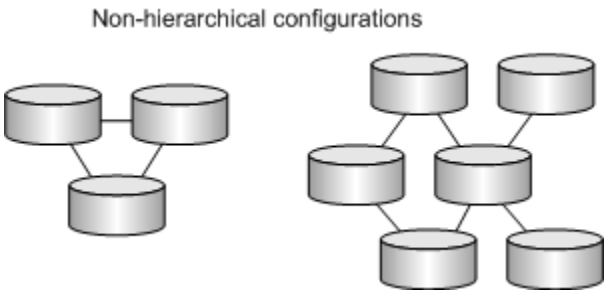
SQL Remote supports hierarchical configurations of databases; it does not support peer-to-peer synchronization or other non-hierarchical configurations. MobiLink is also normally used with a hierarchical configuration, but can also be used in other configurations.

For any two databases directly sharing data in a hierarchical configuration, one is always above or below the other in the hierarchy.

Hierarchical database configurations



Databases in a non-hierarchical configuration do not have a well-defined notion of above or below.



In a MobiLink or SQL Remote system, each database contains all or a subset of the data replicated by the database above it in the hierarchy.

Remote databases can contain tables that are not present at the consolidated database as long as they are not involved in synchronization. SQL Remote requires that the table and column names in the remote databases match the ones in the consolidated database. In contrast, MobiLink allows data to be stored in different columns and tables in the remote databases than in the consolidated database, allowing greater flexibility.

Two-way synchronization

All SQL Anywhere synchronization technologies provide two-way synchronization. Changes made at the central database are propagated to remote databases. Changes made at remote databases are propagated to the central database, and to other remote databases. MobiLink allows upload-only, download-only, and two-way synchronization.

Both SQL Remote and MobiLink allow the same data to be changed simultaneously at multiple locations and both provide a means of resolving any conflicts.

Propagation methods

When a transaction modifies shared data at any one database, the transaction or changes must be replicated to the other databases in the synchronization system. There are various means by which this task may be accomplished.

Session-based synchronization: MobiLink

In a **session-based** or **synchronous** synchronization scheme, synchronization occurs in real time over some sort of direct communications link. For example, the connection can be over a modem, network, or radio modem. Remote sites connect at intervals of seconds, minutes, hours, days, or weeks.

A session-based synchronization process is analogous to a telephone conversation in which all outstanding issues at both ends are resolved. The process follows a particular format. A MobiLink remote site begins by opening a connection to a MobiLink server and uploading a complete list of all the changes made to the remote database since the previous synchronization. Upon receiving this data, the server updates the central

database, and then sends back all relevant changes. The remote site incorporates the entire set of changes, then sends back a confirmation and closes the connection.

Message-based synchronization: SQL Remote

SQL Remote is an **asynchronous** synchronization scheme: it uses messages to exchange data between databases. Messages are typically files or specially formatted email messages. A **message agent**, attached to each database, sends messages regarding changes to its own data. The same agent also receives messages from one or more other databases and modifies the database according to the contents of the received messages.

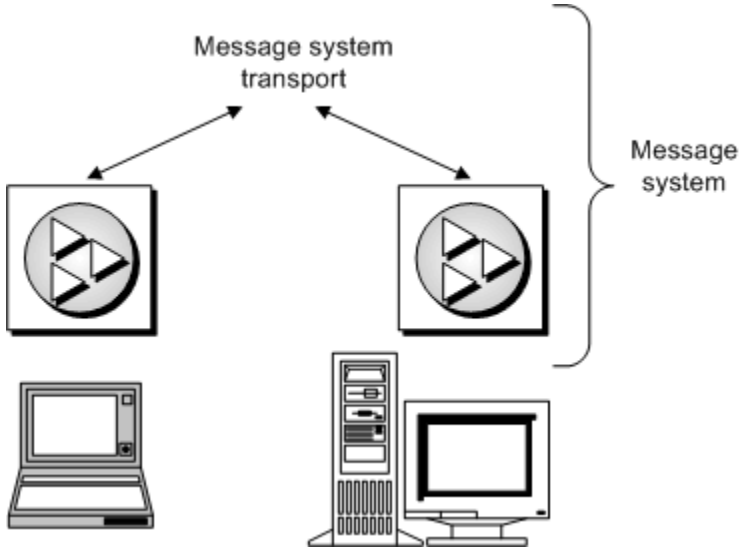
In message-based communications, each message carries its destination address and other control information so that no direct connection is necessary between applications exchanging information. For example, an email message contains the destination address; there is no direct connection between the sending server and the recipient.

Message services use store-and-forward methods

Just as session-based client/server applications rely on network communication protocol stacks, such as TCP/IP, so message-based applications rely on message services such as Internet Simple Mail Transfer Protocol (SMTP), or a simple shared file link.

Message services use **store-and-forward** methods to get each message to its destination: for example, email systems store messages until the recipient opens their mail folder to read their mail, then the email system forwards the message.

Building a synchronization system on top of a message system means that a message-based synchronization system, such as SQL Remote, does not need to implement a store-and-forward system to get messages to their destination. Just as session-based client/server applications do not implement their own protocol stacks to pass information between client and server, so SQL Remote uses existing message systems to pass the messages.



Guaranteed delivery

To work reliably, a message-based synchronization system must both guarantee that all messages reach their destination and that the messages are applied in the same order that they are sent. SQL Remote incorporates a protocol to guarantee application of synchronization updates in the correct order.

Connection-based synchronization

Some synchronization technologies rely on the presence of a continuous, or at least almost continuous, connection between the databases. Through this connection, the two databases conduct an ongoing dialog. These types of systems excel at replicating changes quickly. Given enough resources and channel capacity, synchronization can occur reliably with a lag time of no more than a few seconds.

The main drawback of this type of system is that a reliable, continuous connection can be expensive to maintain. This restriction makes connection-based technologies suited to replication between two large, fixed databases. In environments where the remote computers are mobile or are only occasionally connected, session-based (MobiLink) or message-based (SQL Remote) technologies provide more flexible solutions.

To create a connection-based synchronization system with SQL Anywhere, you need to use Sybase Replication Server. See "Replicating data with Replication Server" [*SQL Anywhere Server - Database Administration*].

Choosing a synchronization technology

Each SQL Anywhere synchronization technology lends itself to particular applications. The following descriptions differentiate between the technologies and let you select the one best suited to your needs.

You should consider which of the following elements are important in your application:

Your consolidated database system

In a typical synchronization environment, a large database serves as a central repository for information. Sometimes you can choose a database system that suits your needs. Other times, a central database already exists and you must adapt the synchronization system to work with it.

MobiLink can work with many popular database servers, including SQL Anywhere, Sybase Adaptive Server Enterprise, Oracle, Microsoft SQL Server, and IBM DB2. Using the MobiLink server APIs for .NET and Java, you can synchronize with any data source, including application servers, web servers, text files, and other database products.

In a SQL Remote system, the central database must be SQL Anywhere.

Your remote database system

SQL Anywhere synchronization technologies also differ in the types of remote databases that they can support.

MobiLink supports SQL Anywhere and UltraLite as remote databases.

SQL Remote supports SQL Anywhere remote databases.

Network characteristics

MobiLink and SQL Remote are both well suited to occasionally-connected environments, where remote sites must operate for hours or days in isolation, although more frequent synchronization is possible whenever a network connection is available.

MobiLink is session-based. A real-time connection is required during synchronization. If this connection is interrupted before synchronization is complete, the process does not complete until the next synchronization. In contrast, SQL Remote relays information via messages, which can be sent or received asynchronously. These messages may take the form of files on a hard disk, or email messages. These messages can be processed whenever they are received, allowing synchronization to occur incrementally.

Frequency of synchronization

In some situations, it may be important that your information is replicated immediately. In others, synchronization once or twice a day may suffice. In fact, more frequent synchronization may be impossible when no network connection is available.

Both MobiLink and SQL Remote are primarily intended for situations where synchronization occurs infrequently, such as every few hours or days, but both can be used to synchronize as frequently as every few seconds.

The number of remote sites

MobiLink and SQL Remote both work well with a very large number of remote users. MobiLink scalability is limited only by the scalability of the consolidated database management system. The SQL Remote message-based design allows a typical installation to handle thousands of remote users.

There is no hard limit on the maximum number of remote sites with any of these systems. The actual number depends on the amount of information replicated, the frequency of synchronization, and the design of your application.

Transaction ordering

By default, MobiLink works by grouping the results of multiple transactions on the remote database into one set of changes to be applied to the consolidated database. Alternatively, you can choose to preserve the order of transactions and upload them separately. In both cases, synchronization always occurs at a transaction boundary, and so referential integrity is preserved. Uncommitted data is never synchronized, and so data integrity is preserved.

SQL Remote replicates data by scanning the transaction log and preparing messages, as appropriate, for each transaction. It orders these messages and sends them to the remote or consolidated site. When processing receives messages, SQL Remote always processes them in the same order as they were applied to the other database. When necessary, it automatically delays processing a message until all earlier messages have been applied.

Achieving data consistency at a particular time

Immediately following each MobiLink synchronization session, the data in the two databases is consistent. The ability to guarantee the consistency of the data at a remote site at a particular point in time is an advantage of MobiLink session-based synchronization. For example, if it is important that the data at a remote site accurately reflect the data in the consolidated database at a particular time, such as 10 o'clock in the morning, this objective can be achieved by synchronizing just prior to this time. As long as the synchronization completes successfully, the currency of the data at the remote site is assured.

When changes to the data are replicated through an exchange of messages, it is difficult to guarantee that the data in a particular remote site is completely consistent with the data in the consolidated site at any particular point in time. For example, sometimes a message is lost in transit. SQL Remote automatically recognizes this fault and resends the message, but such interruptions can cause unexpected delays.

Mobile enterprise messaging: QAnywhere

QAnywhere extends enterprise messaging to mobile applications. Enterprise messaging is a popular and efficient method of exchanging data between business applications. QAnywhere integrates with your enterprise messaging system to provide messaging among mobile devices and between mobile devices and the enterprise. QAnywhere is a comprehensive store and forward messaging solution that connects and integrates information in heterogeneous mobile environments.

QAnywhere provides secure, assured, message delivery for remote and mobile applications. Because QAnywhere automatically handles the challenges of slow and unreliable networks, you can concentrate on application functionality instead of issues surrounding connectivity, communication, and security. QAnywhere store and forward technology ensures that your applications are always available, even when a network connection is not.

QAnywhere is based on proven MobiLink synchronization technology, and has a small footprint and low setup and administration requirements. In addition, the common infrastructure for both data synchronization and messaging, can greatly reduce your administration requirements and simplify deployment.

QAnywhere includes the following characteristics:

- Comprehensive messaging interface with a powerful and flexible programming model for building mobile messaging applications.
- Connectors to back-end JMS-based enterprise systems.
- Reliable and efficient message delivery with compression and transactional capabilities.
- Secure message storage and transmission using 128-bit encryption.
- Network-independent communication.
- Push notification of messages waiting to be delivered.
- Graphical administration tool.

When to use QAnywhere

Use QAnywhere to:

- **Extend back-end enterprise application servers and messaging systems to mobile applications** Use QAnywhere to develop mobile applications that easily integrate with your back-end systems that support the Java Messaging Service (JMS).
- **Add mobile enterprise messaging to an existing data synchronization system** QAnywhere is based on MobiLink synchronization technology, so it is easy to integrate both products in one system. In addition, the common infrastructure greatly reduces administration requirements and simplifies deployment.

  QAnywhere with MobiLink can be combined in multiple ways to control the movement and modification of your data. For example, you can synchronize data to a remote application, use this data to create an order, and then send a message to a middle-tier business logic application for processing.

- **Provide network-independent communication** QAnywhere messages are independent of the network protocol, and can be received by an application that communicates over a different network protocol.
- **Provide communication in occasionally-connected environments** The store-and-forward nature of messaging means that messages can be sent even when the destination application is not currently reachable over the network; the message is delivered when the network becomes available.
- **Use rules-based conditional message transmissions** QAnywhere allows the specification of rules that determine when messages are transmitted and when messages are delivered. These rules can include message properties and network transmission costs.
- **Create a mobile web service** Mobile web services use QAnywhere technology to extend web services to the mobile environment. See "Mobile web services" on page 50.

  See QAnywhere.

Mobile web services

Web services

Web services allow applications running in heterogeneous platforms and languages to interact and exchange data with each other. In a web services system, each application uses an interface to translate the information from the application to the web server. For example, SQL Anywhere can send and receive web service requests through its own built-in web services server or through external web servers. This functionality enables other applications to access information stored inside SQL Anywhere databases. Web services are also used in service-oriented architectures (SOAs).

Mobile web services

SQL Anywhere mobile web services extend your web services to mobile environments. Mobile web services combine the functionality and benefits of web services with SQL Anywhere leading mobile technology. With mobile web services, you can use mobile applications to make web service requests—even when the applications are offline—and have those requests queued for transmission later. Mobile web services use QAnywhere messaging technology to assure delivery of the requests and responses. This means that you can concentrate on developing and accessing the web service as you would in a connected environment. QAnywhere simplifies the transmission, authentication, and serialization of requests and responses in mobile environments.

In addition, mobile web services include the following characteristics:

- A web services connector that supports both HTTP and HTTPS for secure communications.
- The ability to generate a proxy class that simplifies development.

When to use mobile web services

Use mobile web services when:

- You want to access a web service from a mobile application.
- A network connection between the various systems is not continually available.

See “Mobile web services” [*QAnywhere*].

Sample Databases

This section describes the schemas of the SQL Anywhere 11 sample databases. Experiment with the sample databases to learn more about SQL Anywhere 11.

SQL Anywhere sample database

Contents

About the sample database

For consistency and simplicity, many of the examples throughout the documentation use the SQL Anywhere sample database, *demo.db*. This file is installed in the SQL Anywhere samples directory: *samples-dir* *demo.db*.

For information about the default location of *samples-dir*, see "Samples directory" [*SQL Anywhere Server - Database Administration*].

The sample database uses the following default user ID and password:

User ID = **DBA**

Password = **sql** (passwords in SQL Anywhere are case sensitive.)

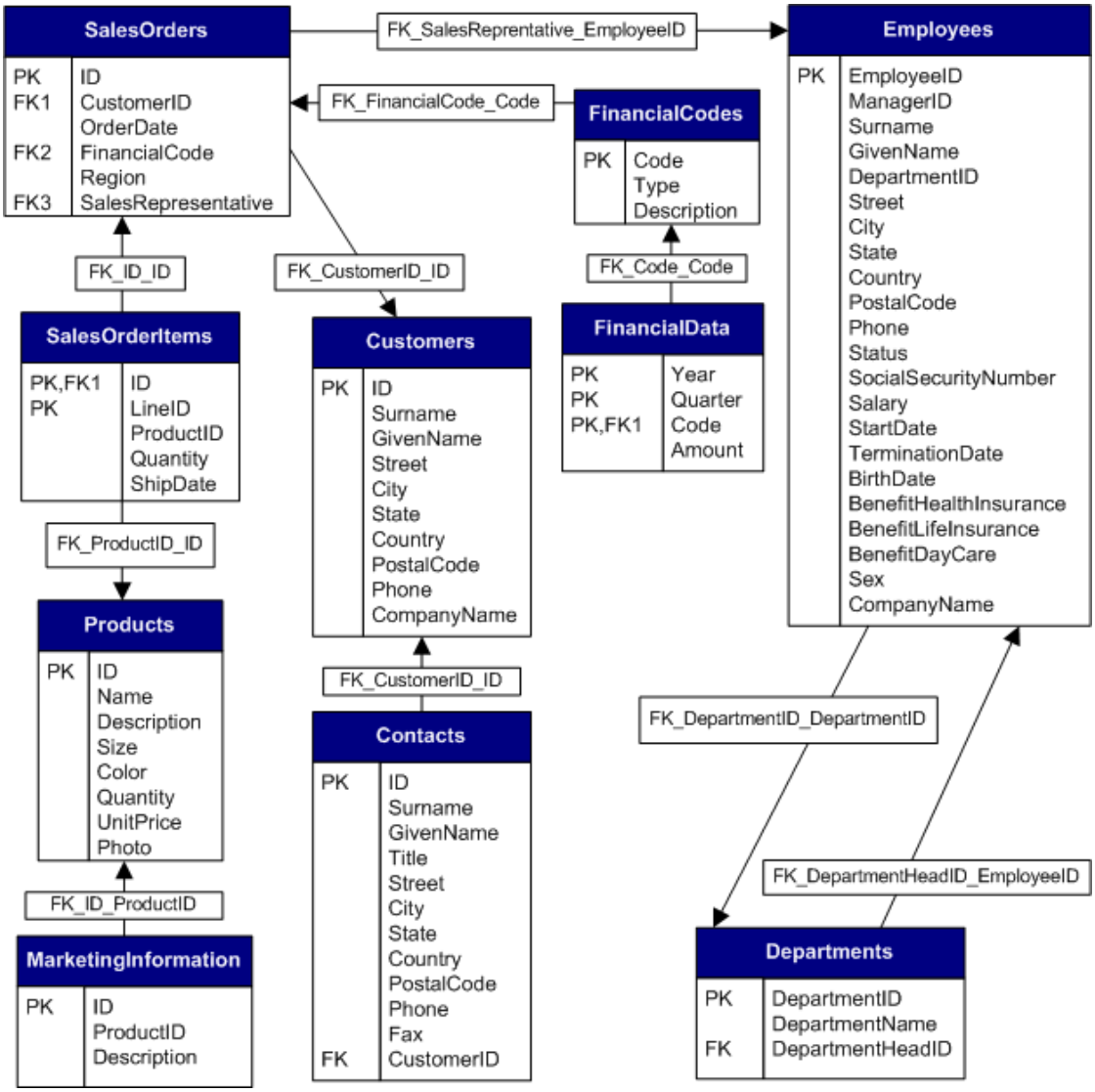
Caution
The sample database always has the same user ID and password; it is recommended that you change the DBA user ID and password to restrict access to the database. See "Changing a password" [*SQL Anywhere Server - Database Administration*].

The sample database uses the following ODBC data source: **SQL Anywhere 11 Demo**.

The sample database represents a small company that sells athletic clothing. It contains internal information about the company (employees, departments, and finances), product information, and sales information (sales orders, customers, and contacts). All data in the database is fictional.

The following figure displays the tables in the sample database and how they are related to each other. The boxes represent tables, and the arrows represent foreign key relationships.

For instructions on how to connect to *demo.db*, see "Tutorial: Using the sample database" [*SQL Anywhere Server - Database Administration*].


SalesOrders
PK ID
FK1 CustomerID
OrderDate
FK2 FinancialCode
Region
FK3 SalesRepresentative
FK_SalesReprentative_EmployeeID
Employees
PK EmployeeID
ManagerID
Surname
GivenName
DepartmentID
Street
City
State
Country
PostalCode
Phone
Status
SocialSecurityNumber
Salary
StartDate
TerminationDate
BirthDate
BenefitHealthInsurance
BenefitLifeInsurance
BenefitDayCare
Sex
CompanyName
FK_FinancialCode_Code
FinancialCodes
PK Code
Type
Description
FK_ID_ID
FK_CustomerID_ID
FK_Code_Code
SalesOrderItems
PK,FK1 ID
PK LineID
ProductID
Quantity
ShipDate
Customers
PK ID
Surname
GivenName
Street
City
State
Country
PostalCode
Phone
CompanyName
FinancialData
PK Year
PK Quarter
PK,FK1 Code
Amount
FK_ProductID_ID
Products
PK ID
Name
Description
Size
Color
Quantity
UnitPrice
Photo
FK_CustomerID_ID
Contacts
PK ID
Surname
GivenName
Title
Street
City
State
Country
PostalCode
Phone
Fax
FK CustomerID
FK_DepartmentID_DepartmentID
FK_ID_ProductID
FK_DepartmentHeadID_EmployeeID
MarketingInformation
PK ID
ProductID
Description
Departments
PK DepartmentID
DepartmentName
FK DepartmentHeadID

Recreate the sample database

Testing features and completing the tutorials in the SQL Anywhere documentation sometimes results in changes to the sample database that can prevent the successful completion of subsequent tutorials and tests. When this happens, you can restore the sample database to its original state. Alternatively, if you need to preserve the sample database in its current state, then you can recreate the sample database in its original state using a different name. Both methods are presented below.

To recreate the sample database

1. At a command prompt, change to the samples directory.

   For the default location of *samples-dir*, see "SQLANYSAMP11 environment variable" [*SQL Anywhere Server - Database Administration*].

2. Execute the following command to erase *demo.db*:

   ```
   dberase demo.db
   ```

3. Type **y** when prompted to confirm your choice and to delete the transaction log.

4. Execute the following command to create a new, empty sample database:

   ```
   dbinit demo.db
   ```

5. At a command prompt, change to the *install-dir**scripts* directory.

   For information about the default location of *install-dir*, see "SQLANY11 environment variable" [*SQL Anywhere Server - Database Administration*].

6. Execute the following command to load the new sample database with objects and data:

   ```
   dbisql -c "DSN=SQL Anywhere 11 Demo" mkdemo.sql
   ```

To recreate the sample database using a different name

1. At a command prompt, change to the samples directory.

   For information about the default location of *samples-dir*, see "SQLANYSAMP11 environment variable" [*SQL Anywhere Server - Database Administration*].

2. Execute the following command to create a database called *mydemo.db*:

   ```
   dbinit mydemo.db
   ```

3. Execute the following command to start the database:

   ```
   dbeng11 mydemo.db
   ```

4. At a command prompt, change to the *install-dir**scripts* directory.

   For the default location of *install-dir*, see "SQLANY11 environment variable" [*SQL Anywhere Server - Database Administration*].

5. Execute the following command to load *mydemo.db* with the objects and data used to create the demo database:


```
dbisql -c "UID=DBA;PWD=sql" mkdemo.sql
```

The CustDB sample database application

Contents

About the CustDB sample database

The CustDB sample application is a useful tool for learning how to develop UltraLite and MobiLink applications. The sample database is a sales status database for a hardware supplier. It holds customer, product, and sales force information for the supplier.

There are two parts to the CustDB sample application:

- **UltraLite** For UltraLite, CustDB can be deployed on any device supported by UltraLite using any platform supported by UltraLite. You can see all the source code used to create the CustDB UltraLite application and run the sample. The CustDB sample application is set up for MobiLink synchronization.

  You can find the UltraLite CustDB sample application in *samples-dir\UltraLite\CustDB*.

  For information about the default location of *samples-dir*, see "Samples directory" [*SQL Anywhere Server - Database Administration*].

  See "UltraLite CustDB samples" [*UltraLite - Database Management and Reference*].

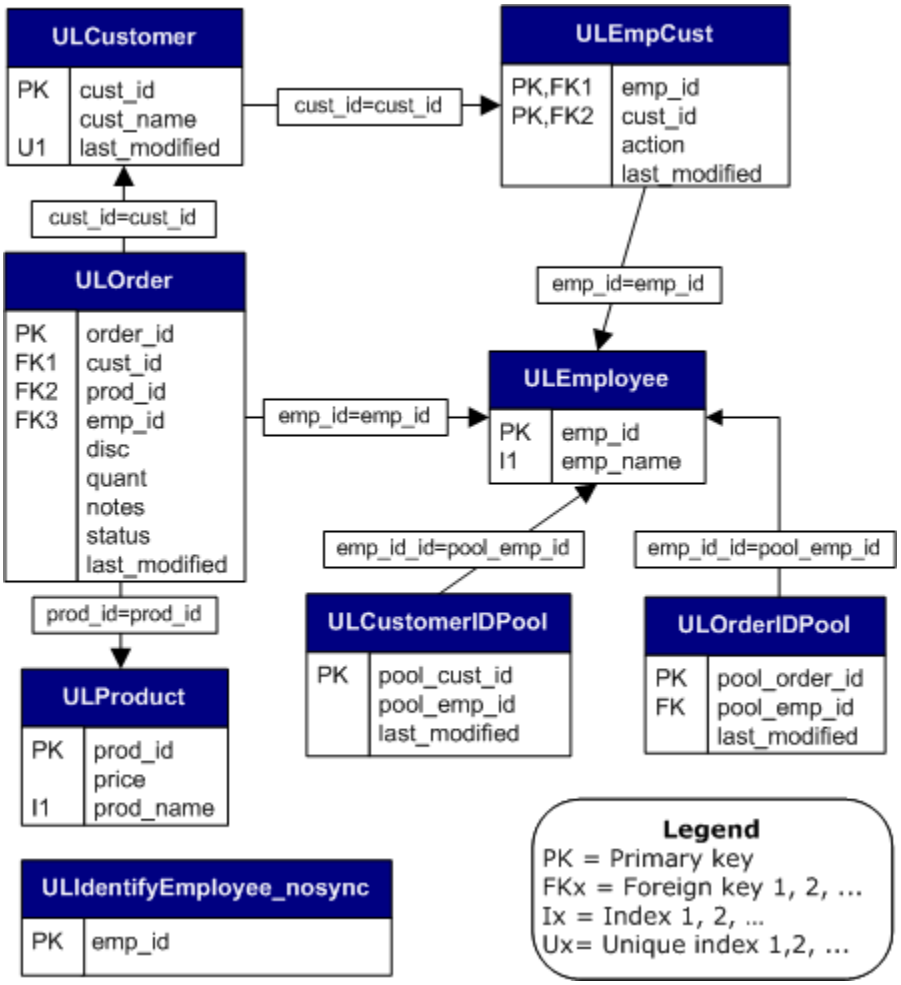
- **MobiLink** If you are interested in exploring MobiLink features, there is a CustDB consolidated database that contains sample synchronization logic. You can use this consolidated database with the CustDB UltraLite sample remote database to run the sample synchronization system.

  The MobiLink consolidated CustDB database is created by running script files against a supported relational database (SQL Anywhere, Adaptive Server Enterprise, Oracle, Microsoft SQL Server, or DB2). These setup files are located in *samples-dir\MobiLink\CustDB*.

  The CustDB consolidated database uses the following ODBC data source: **SQL Anywhere 11 CustDB**.

  See "Exploring the CustDB sample for MobiLink" [*MobiLink - Getting Started*].

The following diagram shows the tables in the CustDB database and how they relate to each other.


ULCustomer
PK cust_id
cust_name
U1 last_modified
cust_id=cust_id
ULEmpCust
PK,FK1 emp_id
PK,FK2 cust_id
action
last_modified
cust_id=cust_id
emp_id=emp_id
ULOrder
PK order_id
FK1 cust_id
FK2 prod_id
FK3 emp_id
disc
quant
notes
status
last_modified
emp_id=emp_id
ULEmployee
PK emp_id
I1 emp_name
emp_id_id=pool_emp_id
emp_id_id=pool_emp_id
prod_id=prod_id
ULCustomerIDPool
PK pool_cust_id
pool_emp_id
last_modified
ULOrderIDPool
PK pool_order_id
FK pool_emp_id
last_modified
ULProduct
PK prod_id
price
I1 prod_name
ULIdentifyEmployee_nosync
PK emp_id
Legend
PK = Primary key
FKx = Foreign key 1, 2, ...
Ix = Index 1, 2, ...
Ux= Unique index 1,2, ...

Getting Started with SQL Anywhere 11

This section provides introductory tutorials for the main SQL Anywhere technologies. It is designed for users who are familiar with databases and who want to start running the software right away.

Getting started with SQL Anywhere 11

Contents

Getting started

Most of the tutorials throughout the documentation use the SQL Anywhere 11 sample database (*demo.db*) or the CustDB sample database application (*custdb.db*).

For information about *demo.db*, see "SQL Anywhere sample database" on page 55.

For information about *custdb.db*, see:

- "Exploring the CustDB sample for MobiLink" [*MobiLink - Getting Started*]
- "UltraLite CustDB samples" [*UltraLite - Database Management and Reference*]

List of getting started

The following is a list of introductory materials and tutorials for the database and data exchange technologies.

- **SQL Anywhere Server** See "Getting started with SQL Anywhere Server" on page 70.

  This section walks you through several tutorials related to connecting to, and administering, SQL Anywhere Server databases.

- **UltraLite** See "UltraLite CustDB samples" [*UltraLite - Database Management and Reference*].

  This chapter uses the UltraLite sample database (*custdb.db*) to demonstrate:

  - Logging in and populating an UltraLite remote database
  - Using a client application
  - Synchronizing an UltraLite database with a consolidated database
  - Browsing MobiLink synchronization scripts

- **MobiLink** See MobiLink - Getting Started.

  This book introduces MobiLink and contains numerous tutorials to guide you through the process of setting up a synchronization system that uses SQL Anywhere databases.

- **QAnywhere** See "Quick start to QAnywhere" [*QAnywhere*].

  This chapter provides an overview of the tasks required to set up and run QAnywhere messaging.

- **SQL Remote** See "SQL Remote introduction" [*SQL Remote*].

  This chapter introduces SQL Remote and describes how to set up a simple SQL Remote replication system using Sybase Central.

Other applications

SQL Anywhere also includes the following applications to create physical data models, reports, and applications:

- **PowerDesigner Physical Data Model** See "About PowerDesigner Physical Data Model" on page 72.

  You can use PowerDesigner Physical Data Model to create physical data models of a database and then convert them into SQL Anywhere databases.

- **InfoMaker** See "About InfoMaker" on page 72.

You can use InfoMaker to create reports quickly.

- **DataWindow .NET** See "About DataWindow .NET" on page 72.

  You can use DataWindow .NET to develop applications.

Getting started with SQL Anywhere Server

Getting started

SQL Anywhere provides three methods to administer your system: Sybase Central, Interactive SQL, and command-line utilities. In most cases, the choice of which tool to use is your preference. The majority of tasks, such as connecting to a database, are supported by all three methods.

The tutorials and introductory materials listed in this section use all three methods to create and manage SQL Anywhere databases.

For an overview of Sybase Central and Interactive SQL, see "Design and management tools" on page 7.

Starting the database server

- "Tutorial: Using the sample database" [*SQL Anywhere Server - Database Administration*]

  This tutorial walks you through starting a database server, displaying the database server messages window, and stopping the database server.

- **Additional information** For a list of the most common options to use when starting a database, or database server, see "Some common options" [*SQL Anywhere Server - Database Administration*].

  For a complete list of server options, see "The SQL Anywhere database server" [*SQL Anywhere Server - Database Administration*].

  Or, run the following command:

  ```
  dbeng11 -?
  ```

- **Further reading** For more information about starting and stopping the database server, see "Running the database server" [*SQL Anywhere Server - Database Administration*].

  For more information about how SQL Anywhere names its databases and servers, see "Naming the server and the databases" [*SQL Anywhere Server - Database Administration*].

Connecting to a database

- "Sample SQL Anywhere database connections" [*SQL Anywhere Server - Database Administration*]

  This section describes how to create an ODBC data source and how to use it to connect to a database using Sybase Central.

- **Further reading** For more examples of connecting to the SQL Anywhere sample database, connecting to an embedded database, and connecting across a network, see "SQL Anywhere database connections" [*SQL Anywhere Server - Database Administration*].

  For more information about ODBC data sources, see "Creating ODBC data sources" [*SQL Anywhere Server - Database Administration*].

Managing databases

- "Using Sybase Central" [*SQL Anywhere Server - Database Administration*]

This section introduces Sybase Central and describes how to use it to:

- Connect to databases
- Create connection profiles
- Search a database
- View entity-relationship diagrams
- Monitor database health and statistics

Creating a database

- "Tutorial: Creating a SQL Anywhere database" [*SQL Anywhere Server - SQL Usage*]

  This tutorial describes how to use Sybase Central to:

  - Create a database file
  - Connect to your database
  - Create a table
  - Create relationships between tables

Using SQL statements

- "Using Interactive SQL" [*SQL Anywhere Server - Database Administration*]

  This section familiarizes you with Interactive SQL and provides steps to:

  - Execute multiple statements
  - Cancel an Interactive SQL command
  - Look up tables, columns, and procedures
  - Print SQL statements
  - Recall and log commands
  - Create and run script files

Other applications

The following sections describe the documentation and resources available for PowerDesigner Physical Data Model, InfoMaker, and DataWindow .NET.

About PowerDesigner Physical Data Model

SQL Anywhere includes Physical Data Model, a module of the powerful Sybase database design tool, PowerDesigner. This module provides ways to generate and modify databases using a graphical representation of the database schema. You can optimize your database by customizing tables, columns, indexes, keys, views, physical storage, triggers, and stored procedures.

PowerDesigner Physical Data Model includes comprehensive documentation, including video tutorials. For more information about PowerDesigner, see www.sybase.com/products/modelingmetadata/powerdesigner.

For information about SQL Anywhere database design, see "Creating databases in SQL Anywhere" [*SQL Anywhere Server - SQL Usage*].

About InfoMaker

InfoMaker is a powerful reporting tool. With InfoMaker, you can create the following objects:

- Reports to view data.
- Forms to view and change data.
- Queries to automatically retrieve data for reports or forms.
- Pipelines to pipe data from one database (or DBMS) to another.
- Applications to bundle reports and forms and distribute them to users.

InfoMaker includes comprehensive documentation. For more information about InfoMaker, see www.sybase.com/products/development/infomaker.

About DataWindow .NET

DataWindow .NET simplifies and accelerates the creation of data-driven applications, whether they are rich client, smart client, mobile or wireless applications. It includes a highly-productive 4GL development environment.

DataWindow .NET includes comprehensive documentation. For more information about DataWindow .NET, see www.sybase.com/products/development/datawindownet.

Glossary

Glossary

Adaptive Server Anywhere (ASA)

The relational database server component of SQL Anywhere Studio, intended for use in mobile and embedded environments or as a server for small and medium-sized businesses. In version 10.0.0, Adaptive Server Anywhere was renamed SQL Anywhere Server, and SQL Anywhere Studio was renamed SQL Anywhere.

See also: "SQL Anywhere" on page 99.

agent ID

See also: "client message store ID" on page 77.

article

In MobiLink or SQL Remote, an article is a database object that represents a whole table, or a subset of the columns and rows in a table. Articles are grouped together in a publication.

See also:

- "replication" on page 97
- "publication" on page 94

atomic transaction

A transaction that is guaranteed to complete successfully or not at all. If an error prevents part of an atomic transaction from completing, the transaction is rolled back to prevent the database from being left in an inconsistent state.

base table

Permanent tables for data. Tables are sometimes called **base tables** to distinguish them from temporary tables and views.

See also:

- "temporary table" on page 101
- "view" on page 103

bit array

A bit array is a type of array data structure that is used for efficient storage of a sequence of bits. A bit array is similar to a character string, except that the individual pieces are 0s (zeros) and 1s (ones) instead of characters. Bit arrays are typically used to hold a string of Boolean values.

business rule

A guideline based on real-world requirements. Business rules are typically implemented through check constraints, user-defined data types, and the appropriate use of transactions.

See also:

- "constraint" on page 79
- "user-defined data type" on page 103

carrier

A MobiLink object, stored in MobiLink system tables or a Notifier properties file, that contains information about a public carrier for use by server-initiated synchronization.

See also: "server-initiated synchronization" on page 98.

character set

A character set is a set of symbols, including letters, digits, spaces, and other symbols. An example of a character set is ISO-8859-1, also known as Latin1.

See also:

- "code page" on page 77
- "encoding" on page 83
- "collation" on page 77

check constraint

A restriction that enforces specified conditions on a column or set of columns.

See also:

- "constraint" on page 79
- "foreign key constraint" on page 84
- "primary key constraint" on page 94
- "unique constraint" on page 102

checkpoint

The point at which all changes to the database are saved to the database file. At other times, committed changes are saved only to the transaction log.

checksum

The calculated number of bits of a database page that is recorded with the database page itself. The checksum allows the database management system to validate the integrity of the page by ensuring that the numbers match as the page is being written to disk. If the counts match, it's assumed that page was successfully written.

client message store

In QAnywhere, a SQL Anywhere database on the remote device that stores messages.

client message store ID

In QAnywhere, a MobiLink remote ID that uniquely identifies a client message store.

client/server

A software architecture where one application (the client) obtains information from and sends information to another application (the server). The two applications often reside on different computers connected by a network.

code page

A code page is an encoding that maps characters of a character set to numeric representations, typically an integer between 0 and 255. An example of a code page is Windows code page 1252. For the purposes of this documentation, code page and encoding are interchangeable terms.

See also:

- “character set” on page 76
- “encoding” on page 83
- “collation” on page 77

collation

A combination of a character set and a sort order that defines the properties of text in the database. For SQL Anywhere databases, the default collation is determined by the operating system and language on which the server is running; for example, the default collation on English Windows systems is 1252LATIN1. A collation, also called a collating sequence, is used for comparing and sorting strings.

See also:

- “character set” on page 76
- “code page” on page 77
- “encoding” on page 83

command file

A text file containing SQL statements. Command files can be built manually, or they can be built automatically by database utilities. The dbunload utility, for example, creates a command file consisting of the SQL statements necessary to recreate a given database.

communication stream

In MobiLink, the network protocol used for communication between the MobiLink client and the MobiLink server.

concurrency

The simultaneous execution of two or more independent, and possibly competing, processes. SQL Anywhere automatically uses locking to isolate transactions and ensure that each concurrent application sees a consistent set of data.

See also:

- "transaction" on page 101
- "isolation level" on page 87

conflict resolution

In MobiLink, conflict resolution is logic that specifies what to do when two users modify the same row on different remote databases.

connection ID

A unique number that identifies a given connection between a client application and the database. You can determine the current connection ID using the following SQL statement:

```
SELECT CONNECTION_PROPERTY( 'Number' );
```

connection-initiated synchronization

A form of MobiLink server-initiated synchronization in which synchronization is initiated when there are changes to connectivity.

See also: "server-initiated synchronization" on page 98.

connection profile

A set of parameters that are required to connect to a database, such as user name, password, and server name, that is stored and used as a convenience.

consolidated database

In distributed database environments, a database that stores the master copy of the data. In case of conflict or discrepancy, the consolidated database is considered to have the primary copy of the data.

See also:

- "synchronization" on page 101
- "replication" on page 97

constraint

A restriction on the values contained in a particular database object, such as a table or column. For example, a column may have a uniqueness constraint, which requires that all values in the column be different. A table may have a foreign key constraint, which specifies how the information in the table relates to data in some other table.

See also:

- "check constraint" on page 76
- "foreign key constraint" on page 84
- "primary key constraint" on page 94
- "unique constraint" on page 102

contention

The act of competing for resources. For example, in database terms, two or more users trying to edit the same row of a database contend for the rights to edit that row.

correlation name

The name of a table or view that is used in the FROM clause of a query—either its original name, or an alternate name, that is defined in the FROM clause.

creator ID

In UltraLite Palm OS applications, an ID that is assigned when the application is created.

cursor

A named linkage to a result set, used to access and update rows from a programming interface. In SQL Anywhere, cursors support forward and backward movement through the query results. Cursors consist of two parts: the cursor result set, typically defined by a SELECT statement; and the cursor position.

See also:

- "cursor result set" on page 79
- "cursor position" on page 79

cursor position

A pointer to one row within the cursor result set.

See also:

- "cursor" on page 79
- "cursor result set" on page 79

cursor result set

The set of rows resulting from a query that is associated with a cursor.

See also:

- "cursor" on page 79
- "cursor position" on page 79

data cube

A multi-dimensional result set with each dimension reflecting a different way to group and sort the same results. Data cubes provide complex information about data that would otherwise require self-join queries and correlated subqueries. Data cubes are a part of OLAP functionality.

data definition language (DDL)

The subset of SQL statements for defining the structure of data in the database. DDL statements create, modify, and remove database objects, such as tables and users.

data manipulation language (DML)

The subset of SQL statements for manipulating data in the database. DML statements retrieve, insert, update, and delete data in the database.

data type

The format of data, such as CHAR or NUMERIC. In the ANSI SQL standard, data types can also include a restriction on size, character set, and collation.

See also: "domain" on page 82.

database

A collection of tables that are related by primary and foreign keys. The tables hold the information in the database. The tables and keys together define the structure of the database. A database management system accesses this information.

See also:

- "foreign key" on page 84
- "primary key" on page 94
- "database management system (DBMS)" on page 81
- "relational database management system (RDBMS)" on page 96

database administrator (DBA)

The user with the permissions required to maintain the database. The DBA is generally responsible for all changes to a database schema, and for managing users and groups. The role of database administrator is automatically built into databases as user ID DBA with password sql.

database connection

A communication channel between a client application and the database. A valid user ID and password are required to establish a connection. The privileges granted to the user ID determine the actions that can be carried out during the connection.

database file

A database is held in one or more database files. There is an initial file, and subsequent files are called dbspaces. Each table, including its indexes, must be contained within a single database file.

See also: "dbspace" on page 82.

database management system (DBMS)

A collection of programs that allow you to create and use databases.

See also: "relational database management system (RDBMS)" on page 96.

database name

The name given to a database when it is loaded by a server. The default database name is the root of the initial database file.

See also: "database file" on page 81.

database object

A component of a database that contains or receives information. Tables, indexes, views, procedures, and triggers are database objects.

database owner (dbo)

A special user that owns the system objects not owned by SYS.

See also:

- "database administrator (DBA)" on page 80
- "SYS" on page 101

database server

A computer program that regulates all access to information in a database. SQL Anywhere provides two types of servers: network servers and personal servers.

DBA authority

The level of permission that enables a user to do administrative activity in the database. The DBA user has DBA authority by default.

See also: "database administrator (DBA)" on page 80.

dbspace

An additional database file that creates more space for data. A database can be held in up to 13 separate files (an initial file and 12 dbspaces). Each table, together with its indexes, must be contained in a single database file. The SQL command CREATE DBSPACE adds a new file to the database.

See also: "database file" on page 81.

deadlock

A state where a set of transactions arrives at a place where none can proceed.

device tracking

In MobiLink server-initiated synchronization, functionality that allows you to address messages using the MobiLink user name that identifies a device.

See also: "server-initiated synchronization" on page 98.

direct row handling

In MobiLink, a way to synchronize table data to sources other than the MobiLink-supported consolidated databases. You can implement both uploads and downloads with direct row handling.

See also:

- "consolidated database" on page 78
- "SQL-based synchronization" on page 99

domain

Aliases for built-in data types, including precision and scale values where applicable, and optionally including DEFAULT values and CHECK conditions. Some domains, such as the monetary data types, are pre-defined in SQL Anywhere. Also called user-defined data type.

See also: "data type" on page 80.

download

The stage in synchronization where data is transferred from the consolidated database to a remote database.

dynamic SQL

SQL that is generated programmatically by your program before it is executed. UltraLite dynamic SQL is a variant designed for small-footprint devices.

EBF

Express Bug Fix. An express bug fix is a subset of the software with one or more bug fixes. The bug fixes are listed in the release notes for the update. Bug fix updates may only be applied to installed software with the same version number. Some testing has been performed on the software, but the software has not

undergone full testing. You should not distribute these files with your application unless you have verified the suitability of the software yourself.

embedded SQL

A programming interface for C programs. SQL Anywhere embedded SQL is an implementation of the ANSI and IBM standard.

encoding

Also known as character encoding, an encoding is a method by which each character in a character set is mapped onto one or more bytes of information, typically represented as a hexadecimal number. An example of an encoding is UTF-8.

See also:

- "character set" on page 76
- "code page" on page 77
- "collation" on page 77

event model

In MobiLink, the sequence of events that make up a synchronization, such as begin_synchronization and download_cursor. Events are invoked if a script is created for them.

external login

An alternate login name and password used when communicating with a remote server. By default, SQL Anywhere uses the names and passwords of its clients whenever it connects to a remote server on behalf of those clients. However, this default can be overridden by creating external logins. External logins are alternate login names and passwords used when communicating with a remote server.

extraction

In SQL Remote replication, the act of unloading the appropriate structure and data from the consolidated database. This information is used to initialize the remote database.

See also: "replication" on page 97.

failover

Switching to a redundant or standby server, system, or network on failure or unplanned termination of the active server, system, or network. Failover happens automatically.

FILE

In SQL Remote replication, a message system that uses shared files for exchanging replication messages. This is useful for testing and for installations without an explicit message-transport system.

See also:"replication" on page 97.

file-based download

In MobiLink, a way to synchronize data in which downloads are distributed as files, allowing offline distribution of synchronization changes.

file-definition database

In MobiLink, a SQL Anywhere database that is used for creating download files.

See also: “file-based download” on page 84.

foreign key

One or more columns in a table that duplicate the primary key values in another table. Foreign keys establish relationships between tables.

See also:

- “primary key” on page 94
- “foreign table” on page 84

foreign key constraint

A restriction on a column or set of columns that specifies how the data in the table relates to the data in some other table. Imposing a foreign key constraint on a set of columns makes those columns the foreign key.

See also:

- “constraint” on page 79
- “check constraint” on page 76
- “primary key constraint” on page 94
- “unique constraint” on page 102

foreign table

The table containing the foreign key.

See also: “foreign key” on page 84.

full backup

A backup of the entire database, and optionally, the transaction log. A full backup contains all the information in the database and provides protection in the event of a system or media failure.

See also: “incremental backup” on page 86.

gateway

A MobiLink object, stored in MobiLink system tables or a Notifier properties file, that contains information about how to send messages for server-initiated synchronization.

See also: “server-initiated synchronization” on page 98.

generated join condition

A restriction on join results that is automatically generated. There are two types: key and natural. Key joins are generated when you specify KEY JOIN or when you specify the keyword JOIN but do not use the keywords CROSS, NATURAL, or ON. For a key join, the generated join condition is based on foreign key relationships between tables. Natural joins are generated when you specify NATURAL JOIN; the generated join condition is based on common column names in the two tables.

See also:

- "join" on page 88
- "join condition" on page 88

generation number

In MobiLink, a mechanism for forcing remote databases to upload data before applying any more download files.

See also: "file-based download" on page 84.

global temporary table

A type of temporary table for which data definitions are visible to all users until explicitly dropped. Global temporary tables let each user open their own identical instance of a table. By default, rows are deleted on commit, and rows are always deleted when the connection is ended.

See also:

- "temporary table" on page 101
- "local temporary table" on page 88

grant option

The level of permission that allows a user to grant permissions to other users.

hash

A hash is an index optimization that transforms index entries into keys. An index hash aims to avoid the expensive operation of finding, loading, and then unpacking the rows to determine the indexed value, by including enough of the actual row data with its row ID.

histogram

The most important component of column statistics, histograms are a representation of data distribution. SQL Anywhere maintains histograms to provide the optimizer with statistical information about the distribution of values in columns.

iAnywhere JDBC driver

The iAnywhere JDBC driver provides a JDBC driver that has some performance benefits and feature benefits compared to the pure Java jConnect JDBC driver, but which is not a pure-Java solution. The iAnywhere JDBC driver is recommended in most cases.

See also:

- "JDBC" on page 87
- "jConnect" on page 87

identifier

A string of characters used to reference a database object, such as a table or column. An identifier may contain any character from A through Z, a through z, 0 through 9, underscore (_), at sign (@), number sign (#), or dollar sign ($).

incremental backup

A backup of the transaction log only, typically used between full backups.

See also: "transaction log" on page 101.

index

A sorted set of keys and pointers associated with one or more columns in a base table. An index on one or more columns of a table can improve performance.

InfoMaker

A reporting and data maintenance tool that lets you create sophisticated forms, reports, graphs, cross-tabs, and tables, and applications that use these reports as building blocks.

inner join

A join in which rows appear in the result set only if both tables satisfy the join condition. Inner joins are the default.

See also:

- "join" on page 88
- "outer join" on page 92

integrated login

A login feature that allows the same single user ID and password to be used for operating system logins, network logins, and database connections.

integrity

Adherence to rules that ensure that data is correct and accurate, and that the relational structure of the database is intact.

See also: "referential integrity" on page 96.

Interactive SQL

A SQL Anywhere application that allows you to query and alter data in your database, and modify the structure of your database. Interactive SQL provides a pane for you to enter SQL statements, and panes that return information about how the query was processed and the result set.

isolation level

The degree to which operations in one transaction are visible to operations in other concurrent transactions. There are four isolation levels, numbered 0 through 3. Level 3 provides the highest level of isolation. Level 0 is the default setting. SQL Anywhere also supports three snapshot isolation levels: snapshot, statement-snapshot, and readonly-statement-snapshot.

See also: "snapshot isolation" on page 99.

JAR file

Java archive file. A compressed file format consisting of a collection of one or more packages used for Java applications. It includes all the resources necessary to install and run a Java program in a single compressed file.

Java class

The main structural unit of code in Java. It is a collection of procedures and variables grouped together because they all relate to a specific, identifiable category.

jConnect

A Java implementation of the JavaSoft JDBC standard. It provides Java developers with native database access in multi-tier and heterogeneous environments. However, the iAnywhere JDBC driver is the preferred JDBC driver for most cases.

See also:

- "JDBC" on page 87
- "iAnywhere JDBC driver" on page 86

JDBC

Java Database Connectivity. A SQL-language programming interface that allows Java applications to access relational data. The preferred JDBC driver is the iAnywhere JDBC driver.

See also:

- "jConnect" on page 87
- "iAnywhere JDBC driver" on page 86

join

A basic operation in a relational system that links the rows in two or more tables by comparing the values in specified columns.

join condition

A restriction that affects join results. You specify a join condition by inserting an ON clause or WHERE clause immediately after the join. In the case of natural and key joins, SQL Anywhere generates a join condition.

See also:

- "join" on page 88
- "generated join condition" on page 85

join type

SQL Anywhere provides four types of joins: cross join, key join, natural join, and joins using an ON clause.

See also: "join" on page 88.

light weight poller

In MobiLink server-initiated synchronization, a device application that polls for push notifications from a MobiLink server.

See also: "server-initiated synchronization" on page 98.

Listener

A program, dblsn, that is used for MobiLink server-initiated synchronization. Listeners are installed on remote devices and configured to initiate actions on the device when they receive push notifications.

See also: "server-initiated synchronization" on page 98.

local temporary table

A type of temporary table that exists only for the duration of a compound statement or until the end of the connection. Local temporary tables are useful when you need to load a set of data only once. By default, rows are deleted on commit.

See also:

- "temporary table" on page 101
- "global temporary table" on page 85

lock

A concurrency control mechanism that protects the integrity of data during the simultaneous execution of multiple transactions. SQL Anywhere automatically applies locks to prevent two connections from changing the same data at the same time, and to prevent other connections from reading data that is in the process of being changed.

You control locking by setting the isolation level.

See also:

- "isolation level" on page 87
- "concurrency" on page 78
- "integrity" on page 87

log file

A log of transactions maintained by SQL Anywhere. The log file is used to ensure that the database is recoverable in the event of a system or media failure, to improve database performance, and to allow data replication using SQL Remote.

See also:

- "transaction log" on page 101
- "transaction log mirror" on page 102
- "full backup" on page 84

logical index

A reference (pointer) to a physical index. There is no indexing structure stored on disk for a logical index.

LTM

Log Transfer Manager (LTM) also called Replication Agent. Used with Replication Server, the LTM is the program that reads a database transaction log and sends committed changes to Sybase Replication Server.

See: "Replication Server" on page 97.

maintenance release

A maintenance release is a complete set of software that upgrades installed software from an older version with the same major version number (version number format is *major*.*minor*.*patch*.*build*). Bug fixes and other changes are listed in the release notes for the upgrade.

materialized view

A materialized view is a view that has been computed and stored on disk. Materialized views have characteristics of both views (they are defined using a query specification), and of tables (they allow most table operations to be performed on them).

See also:

- "base table" on page 75
- "view" on page 103

message log

A log where messages from an application such as a database server or MobiLink server can be stored. This information can also appear in a messages window or be logged to a file. The message log includes informational messages, errors, warnings, and messages from the MESSAGE statement.

message store

In QAnywhere, databases on the client and server device that store messages.

See also:

- "client message store" on page 77
- "server message store" on page 99

message system

In SQL Remote replication, a protocol for exchanging messages between the consolidated database and a remote database. SQL Anywhere includes support for the following message systems: FILE, FTP, and SMTP.

See also:

- "replication" on page 97
- "FILE" on page 83

message type

In SQL Remote replication, a database object that specifies how remote users communicate with the publisher of a consolidated database. A consolidated database may have several message types defined for it; this allows different remote users to communicate with it using different message systems.

See also:

- "replication" on page 97
- "consolidated database" on page 78

metadata

Data about data. Metadata describes the nature and content of other data.

See also: "schema" on page 98.

mirror log

See also: "transaction log mirror" on page 102.

MobiLink

A session-based synchronization technology designed to synchronize UltraLite and SQL Anywhere remote databases with a consolidated database.

See also:

- “consolidated database” on page 78
- “synchronization” on page 101
- “UltraLite” on page 102

MobiLink client

There are two kinds of MobiLink clients. For SQL Anywhere remote databases, the MobiLink client is the dbmlsync command line utility. For UltraLite remote databases, the MobiLink client is built in to the UltraLite runtime library.

MobiLink Monitor

A graphical tool for monitoring MobiLink synchronizations.

MobiLink server

The computer program that runs MobiLink synchronization, mlsrv11.

MobiLink system table

System tables that are required by MobiLink synchronization. They are installed by MobiLink setup scripts into the MobiLink consolidated database.

MobiLink user

A MobiLink user is used to connect to the MobiLink server. You create the MobiLink user on the remote database and register it in the consolidated database. MobiLink user names are entirely independent of database user names.

network protocol

The type of communication, such as TCP/IP or HTTP.

network server

A database server that accepts connections from computers sharing a common network.

See also: “personal server” on page 93.

normalization

The refinement of a database schema to eliminate redundancy and improve organization according to rules based on relational database theory.

Notifier

A program that is used by MobiLink server-initiated synchronization. Notifiers are integrated into the MobiLink server. They check the consolidated database for push requests, and send push notifications.

See also:

- "server-initiated synchronization" on page 98
- "Listener" on page 88

object tree

In Sybase Central, the hierarchy of database objects. The top level of the object tree shows all products that your version of Sybase Central supports. Each product expands to reveal its own sub-tree of objects.

See also: "Sybase Central" on page 100.

ODBC

Open Database Connectivity. A standard Windows interface to database management systems. ODBC is one of several interfaces supported by SQL Anywhere.

ODBC Administrator

A Microsoft program included with Windows operating systems for setting up ODBC data sources.

ODBC data source

A specification of the data a user wants to access via ODBC, and the information needed to get to that data.

outer join

A join that preserves all the rows in a table. SQL Anywhere supports left, right, and full outer joins. A left outer join preserves the rows in the table to the left of the join operator, and returns a null when a row in the right table does not satisfy the join condition. A full outer join preserves all the rows from both tables.

See also:

- "join" on page 88
- "inner join" on page 86

package

In Java, a collection of related classes.

parse tree

An algebraic representation of a query.

PDB

A Palm database file.

performance statistic

A value reflecting the performance of the database system. The CURRREAD statistic, for example, represents the number of file reads issued by the database server that have not yet completed.

personal server

A database server that runs on the same computer as the client application. A personal database server is typically used by a single user on a single computer, but it can support several concurrent connections from that user.

physical index

The actual indexing structure of an index, as it is stored on disk.

plug-in module

In Sybase Central, a way to access and administer a product. Plug-ins are usually installed and registered automatically with Sybase Central when you install the respective product. Typically, a plug-in appears as a top-level container, in the Sybase Central main window, using the name of the product itself; for example, SQL Anywhere.

See also: "Sybase Central" on page 100.

policy

In QAnywhere, the way you specify when message transmission should occur.

polling

In MobiLink server-initiated synchronization, the way a light weight poller, such as the MobiLink Listener, requests push notifications from a Notifier.

See also: "server-initiated synchronization" on page 98.

PowerDesigner

A database modeling application. PowerDesigner provides a structured approach to designing a database or data warehouse. SQL Anywhere includes the Physical Data Model component of PowerDesigner.

PowerJ

A Sybase product for developing Java applications.

predicate

A conditional expression that is optionally combined with the logical operators AND and OR to make up the set of conditions in a WHERE or HAVING clause. In SQL, a predicate that evaluates to UNKNOWN is interpreted as FALSE.

primary key

A column or list of columns whose values uniquely identify every row in the table.

See also: “foreign key” on page 84.

primary key constraint

A uniqueness constraint on the primary key columns. A table can have only one primary key constraint.

See also:

- “constraint” on page 79
- “check constraint” on page 76
- “foreign key constraint” on page 84
- “unique constraint” on page 102
- “integrity” on page 87

primary table

The table containing the primary key in a foreign key relationship.

proxy table

A local table containing metadata used to access a table on a remote database server as if it were a local table.

See also: “metadata” on page 90.

publication

In MobiLink or SQL Remote, a database object that identifies data that is to be synchronized. In MobiLink, publications exist only on the clients. A publication consists of articles. SQL Remote users can receive a publication by subscribing to it. MobiLink users can synchronize a publication by creating a synchronization subscription to it.

See also:

- “replication” on page 97
- “article” on page 75
- “publication update” on page 94

publication update

In SQL Remote replication, a list of changes made to one or more publications in one database. A publication update is sent periodically as part of a replication message to the remote database(s).

See also:

- “replication” on page 97
- “publication” on page 94

publisher

In SQL Remote replication, the single user in a database who can exchange replication messages with other replicating databases.

See also: "replication" on page 97.

push notification

In QAnywhere, a special message delivered from the server to a QAnywhere client that prompts the client to initiate a message transmission. In MobiLink server-initiated synchronization, a special message delivered from a Notifer to a device that contains push request data and internal information.

See also:

- "QAnywhere" on page 95
- "server-initiated synchronization" on page 98

push request

In MobiLink server-initiated synchronization, a row of values in a result set that a Notifier checks to determine if push notifications need to be sent to a device.

See also: "server-initiated synchronization" on page 98.

QAnywhere

Application-to-application messaging, including mobile device to mobile device and mobile device to and from the enterprise, that permits communication between custom programs running on mobile or wireless devices and a centrally located server application.

QAnywhere agent

In QAnywhere, a process running on the client device that monitors the client message store and determines when message transmission should occur.

query

A SQL statement or group of SQL statements that access and/or manipulate data in a database.

See also: "SQL" on page 99.

Redirector

A web server plug-in that routes requests and responses between a client and the MobiLink server. This plug-in also implements load-balancing and failover mechanisms.

reference database

In MobiLink, a SQL Anywhere database used in the development of UltraLite clients. You can use a single SQL Anywhere database as both reference and consolidated database during development. Databases made with other products cannot be used as reference databases.

referencing object

An object, such as a view, whose definition directly references another object in the database, such as a table.

See also: “foreign key” on page 84.

referenced object

An object, such as a table, that is directly referenced in the definition of another object, such as a view.

See also: “primary key” on page 94.

referential integrity

Adherence to rules governing data consistency, specifically the relationships between the primary and foreign key values in different tables. To have referential integrity, the values in each foreign key must correspond to the primary key values of a row in the referenced table.

See also:

- “primary key” on page 94
- “foreign key” on page 84

regular expression

A regular expression is a sequence of characters, wildcards, and operators that defines a pattern to search for within a string.

relational database management system (RDBMS)

A type of database management system that stores data in the form of related tables.

See also: “database management system (DBMS)” on page 81.

remote database

In MobiLink or SQL Remote, a database that exchanges data with a consolidated database. Remote databases may share all or some of the data in the consolidated database.

See also:

- “synchronization” on page 101
- “consolidated database” on page 78

REMOTE DBA authority

In SQL Remote, a level of permission required by the Message Agent (dbremote). In MobiLink, a level of permission required by the SQL Anywhere synchronization client (dbmlsync). When the Message Agent (dbremote) or synchronization client connects as a user who has this authority, it has full DBA access. The user ID has no additional permissions when not connected through the Message Agent (dbremote) or synchronization client (dbmlsync).

See also: “DBA authority” on page 81.

remote ID

A unique identifier in SQL Anywhere and UltraLite databases that is used by MobiLink. The remote ID is initially set to NULL and is set to a GUID during a database's first synchronization.

replication

The sharing of data among physically distinct databases. Sybase has three replication technologies: MobiLink, SQL Remote, and Replication Server.

Replication Agent

See: "LTM" on page 89.

replication frequency

In SQL Remote replication, a setting for each remote user that determines how often the publisher's message agent should send replication messages to that remote user.

See also: "replication" on page 97.

replication message

In SQL Remote or Replication Server, a communication sent between a publishing database and a subscribing database. Messages contain data, passthrough statements, and information required by the replication system.

See also:

- "replication" on page 97
- "publication update" on page 94

Replication Server

A Sybase connection-based replication technology that works with SQL Anywhere and Adaptive Server Enterprise. It is intended for near-real time replication between a few databases.

See also: "LTM" on page 89.

role

In conceptual database modeling, a verb or phrase that describes a relationship from one point of view. You can describe each relationship with two roles. Examples of roles are "contains" and "is a member of."

role name

The name of a foreign key. This is called a role name because it names the relationship between the foreign table and primary table. By default, the role name is the table name, unless another foreign key is already using that name, in which case the default role name is the table name followed by a three-digit unique number. You can also create the role name yourself.

See also: "foreign key" on page 84.

rollback log

A record of the changes made during each uncommitted transaction. In the event of a ROLLBACK request or a system failure, uncommitted transactions are reversed out of the database, returning the database to its former state. Each transaction has a separate rollback log, which is deleted when the transaction is complete.

See also: "transaction" on page 101.

row-level trigger

A trigger that executes once for each row that is changed.

See also:

- "trigger" on page 102
- "statement-level trigger" on page 100

schema

The structure of a database, including tables, columns, and indexes, and the relationships between them.

script

In MobiLink, code written to handle MobiLink events. Scripts programmatically control data exchange to meet business needs.

See also: "event model" on page 83.

script-based upload

In MobiLink, a way to customize the upload process as an alternative to using the log file.

script version

In MobiLink, a set of synchronization scripts that are applied together to create a synchronization.

secured feature

A feature specified by the -sf option when a database server is started, so it is not available for any database running on that database server.

server-initiated synchronization

A way to initiate MobiLink synchronization from the MobiLink server.

server management request

A QAnywhere message that is formatted as XML and sent to the QAnywhere system queue as a way to administer the server message store or monitor QAnywhere applications.

server message store

In QAnywhere, a relational database on the server that temporarily stores messages until they are transmitted to a client message store or JMS system. Messages are exchanged between clients via the server message store.

service

In Windows operating systems, a way of running applications when the user ID running the application is not logged on.

session-based synchronization

A type of synchronization where synchronization results in consistent data representation across both the consolidated and remote databases. MobiLink is session-based.

snapshot isolation

A type of isolation level that returns a committed version of the data for transactions that issue read requests. SQL Anywhere provides three snapshot isolation levels: snapshot, statement-snapshot, and readonly-statement-snapshot. When using snapshot isolation, read operations do not block write operations.

See also: "isolation level" on page 87.

SQL

The language used to communicate with relational databases. ANSI has defined standards for SQL, the latest of which is SQL-2003. SQL stands, unofficially, for Structured Query Language.

SQL Anywhere

The relational database server component of SQL Anywhere that is intended for use in mobile and embedded environments or as a server for small and medium-sized businesses. SQL Anywhere is also the name of the package that contains the SQL Anywhere RDBMS, the UltraLite RDBMS, MobiLink synchronization software, and other components.

SQL-based synchronization

In MobiLink, a way to synchronize table data to MobiLink-supported consolidated databases using MobiLink events. For SQL-based synchronization, you can use SQL directly or you can return SQL using the MobiLink server APIs for Java and .NET.

SQL Remote

A message-based data replication technology for two-way replication between consolidated and remote databases. The consolidated and remote databases must be SQL Anywhere.

SQL statement

A string containing SQL keywords designed for passing instructions to a DBMS.

See also:

- "schema" on page 98
- "SQL" on page 99
- "database management system (DBMS)" on page 81

statement-level trigger

A trigger that executes after the entire triggering statement is completed.

See also:

- "trigger" on page 102
- "row-level trigger" on page 98

stored procedure

A stored procedure is a group of SQL instructions stored in the database and used to execute a set of operations or queries on a database server

string literal

A string literal is a sequence of characters enclosed in single quotes.

subquery

A SELECT statement that is nested inside another SELECT, INSERT, UPDATE, or DELETE statement, or another subquery.

There are two types of subquery: correlated and nested.

subscription

In MobiLink synchronization, a link in a client database between a publication and a MobiLink user, allowing the data described by the publication to be synchronized.

In SQL Remote replication, a link between a publication and a remote user, allowing the user to exchange updates on that publication with the consolidated database.

See also:

- "publication" on page 94
- "MobiLink user" on page 91

Sybase Central

A database management tool that provides SQL Anywhere database settings, properties, and utilities in a graphical user interface. Sybase Central can also be used for managing other Sybase products, including MobiLink.

synchronization

The process of replicating data between databases using MobiLink technology.

In SQL Remote, synchronization is used exclusively to denote the process of initializing a remote database with an initial set of data.

See also:

- "MobiLink" on page 91
- "SQL Remote" on page 99

SYS

A special user that owns most of the system objects. You cannot log in as SYS.

system object

Database objects owned by SYS or dbo.

system table

A table, owned by SYS or dbo, that holds metadata. System tables, also known as data dictionary tables, are created and maintained by the database server.

system view

A type of view, included in every database, that presents the information held in the system tables in an easily understood format.

temporary table

A table that is created for the temporary storage of data. There are two types: global and local.

See also:

- "local temporary table" on page 88
- "global temporary table" on page 85

transaction

A sequence of SQL statements that comprise a logical unit of work. A transaction is processed in its entirety or not at all. SQL Anywhere supports transaction processing, with locking features built in to allow concurrent transactions to access the database without corrupting the data. Transactions end either with a COMMIT statement, which makes the changes to the data permanent, or a ROLLBACK statement, which undoes all the changes made during the transaction.

transaction log

A file storing all changes made to a database, in the order in which they are made. It improves performance and allows data recovery in the event the database file is damaged.

transaction log mirror

An optional identical copy of the transaction log file, maintained simultaneously. Every time a database change is written to the transaction log file, it is also written to the transaction log mirror file.

A mirror file should be kept on a separate device from the transaction log, so that if either device fails, the other copy of the log keeps the data safe for recovery.

See also: "transaction log" on page 101.

transactional integrity

In MobiLink, the guaranteed maintenance of transactions across the synchronization system. Either a complete transaction is synchronized, or no part of the transaction is synchronized.

transmission rule

In QAnywhere, logic that determines when message transmission is to occur, which messages to transmit, and when messages should be deleted.

trigger

A special form of stored procedure that is executed automatically when a user runs a query that modifies the data.

See also:

- "row-level trigger" on page 98
- "statement-level trigger" on page 100
- "integrity" on page 87

UltraLite

A database optimized for small, mobile, and embedded devices. Intended platforms include cell phones, pagers, and personal organizers.

UltraLite runtime

An in-process relational database management system that includes a built-in MobiLink synchronization client. The UltraLite runtime is included in the libraries used by each of the UltraLite programming interfaces, and in the UltraLite engine.

unique constraint

A restriction on a column or set of columns requiring that all non-null values are different. A table can have multiple unique constraints.

See also:

- "foreign key constraint" on page 84
- "primary key constraint" on page 94
- "constraint" on page 79

unload

Unloading a database exports the structure and/or data of the database to text files (SQL command files for the structure, and ASCII comma-separated files for the data). You unload a database with the Unload utility.

In addition, you can unload selected portions of your data using the UNLOAD statement.

upload

The stage in synchronization where data is transferred from a remote database to a consolidated database.

user-defined data type

See "domain" on page 82.

validate

To test for particular types of file corruption of a database, table, or index.

view

A SELECT statement that is stored in the database as an object. It allows users to see a subset of rows or columns from one or more tables. Each time a user uses a view of a particular table, or combination of tables, it is recomputed from the information stored in those tables. Views are useful for security purposes, and to tailor the appearance of database information to make data access straightforward.

window

The group of rows over which an analytic function is performed. A window may contain one, many, or all rows of data that has been partitioned according to the grouping specifications provided in the window definition. The window moves to include the number or range of rows needed to perform the calculations for the current row in the input. The main benefit of the window construct is that it allows additional opportunities for grouping and analysis of results, without having to perform additional queries.

Windows

The Microsoft Windows family of operating systems, such as Windows Vista, Windows XP, and Windows 200x.

Windows CE

See "Windows Mobile" on page 103.

Windows Mobile

A family of operating systems produced by Microsoft for mobile devices.

work table

An internal storage area for interim results during query optimization.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X