SYBASE iAnywhere

SQL Anywhere® 12
Introduction

Contents

About this book

This book introduces SQL Anywhere 12, 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.

About the SQL Anywhere documentation

The complete SQL Anywhere documentation is available in four formats:

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

  To access the documentation, go to http://dcx.sybase.com.

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

  To access the documentation, choose **Start** » **Programs** » **SQL Anywhere 12** » **Documentation** » **HTML Help (English)**.

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

- **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 access the PDF documentation on Windows operating systems, choose **Start** » **Programs** » **SQL Anywhere 12** » **Documentation** » **PDF (English)**.

  To access the PDF documentation on Unix operating systems, use a web browser to open */documentation/en/pdf/index.html* under the SQL Anywhere installation directory.

Documentation conventions

This section lists the conventions used in this documentation.

Operating systems

SQL Anywhere runs on a variety of platforms. Typically, the behavior of the software is 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 (IBM 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 platforms that are used primarily on server, desktop, and laptop computers, as well as platforms used on mobile devices. Unless otherwise specified, when the documentation refers to Windows, it refers to all supported Windows-based platforms, including Windows Mobile.

  Windows Mobile is based on the Windows CE operating system, which is also used to build a variety of platforms other than Windows Mobile. Unless otherwise specified, when the documentation refers to Windows Mobile, it refers to all supported platforms built using Windows CE.

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

For the complete list of platforms supported by SQL Anywhere, see "Supported platforms" on page 6.

Directory and file names

Usually 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. You can usually 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 *dbsrv12.exe*. On Unix, it is *dbsrv12*.

- ***install-dir*** During the installation process, you choose where to install SQL Anywhere. The environment variable SQLANY12 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 *%SQLANY12%\readme.txt*. On Unix, this is equivalent to *$SQLANY12/readme.txt* or *${SQLANY12}/readme.txt*.

  For more information about the default location of *install-dir*, see "SQLANY12 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 SQLANYSAMP12 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*, choose **Start** » **Programs** » **SQL Anywhere 12** » **Sample Applications And Projects**.

  For more information about the default location of *samples-dir*, see "SQLANYSAMP12 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)"
  ```

- **Semicolons** On Unix, semicolons should be enclosed in quotes.
- **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}*.

Contacting the documentation team

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

You can 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

Go to http://dcx.sybase.com.

Finding out more and requesting technical support

Newsgroups

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: **dbeng12 -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.

Developer Centers

The **SQL Anywhere Tech Corner** gives developers easy access to product technical documentation. You can browse technical white papers, FAQs, tech notes, downloads, techcasts and more to find answers to your questions as well as solutions to many common issues. See http://www.sybase.com/developer/library/sql-anywhere-techcorner.

The following table contains a list of the developer centers available for use on the SQL Anywhere Tech Corner:

Name	URL	Description
SQL Anywhere .NET Developer Center	www.sybase.com/developer/library/sql-anywhere-techcorner/microsoft-net	Get started and get answers to specific questions regarding SQL Anywhere and .NET development.
PHP Developer Center	www.sybase.com/developer/library/sql-anywhere-techcorner/php	An introduction to using the PHP (PHP Hypertext Preprocessor) scripting language to query your SQL Anywhere database.

Name	URL	Description
SQL Anywhere Windows Mobile Developer Center	www.sybase.com/developer/library/sql-anywhere-techcorner/windows-mobile	Get started and get answers to specific questions regarding SQL Anywhere and Windows Mobile development.

SQL Anywhere 12 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 concepts" on page 10.

- **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 "Overview of data exchange technologies" on page 27.

- **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 39.

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" on page 20.

- **Desktop applications** SQL Anywhere delivers enterprise-caliber features, 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 19.

- **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 30.

- **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 also:

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

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, see http://www.sybase.com/detail?id=1068247 .

For information on separately licensed components, see "Separately licensed components" on page 3.

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.

For more information about licensing, see "Server Licensing utility (dblic)" [*SQL Anywhere Server - Database Administration*] and http://www.sybase.com/detail?id=1056242.

See also

- "-gt dbeng12/dbsrv12 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 go to 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 2.

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:

Feature	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 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 dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere read-only scale-out option

For SQL Anywhere databases, the license for using read-only scale-out must be ordered separately. Read-only scale-out is a configuration that allows you to offload reporting or other operations that require read-only access to the database. For more information about the read-only scale-out option, see "SQL Anywhere read-only scale-out" [*SQL Anywhere Server - Database Administration*].

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*].

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 failover for server initiated sync.

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

SQL Anywhere Monitor Production Edition

The SQL Anywhere Monitor Production Edition must be ordered separately. The production edition is intended for deployment and production use. For more information about the Monitor, see "SQL Anywhere Monitor" [*SQL Anywhere Server - Database Administration*].

Hallmarks of SQL Anywhere 12

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 Mac OS X. 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 SE 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, Sybase 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 3.

Supported platforms

The **SQL Anywhere Supported Platforms and Engineering Support Status** web page lists the supported operating system platforms broken down by version of SQL Anywhere. It also indicates the engineering support status for each SQL Anywhere version. See http://www.sybase.com/detail?id=1002288.

From this web page, you can go to the following web pages for more support information, such as:

Web page name	URL	Description
SQL Anywhere Components by Platforms	http://www.sybase.com/detail?id=1061806	Provides a list of components that are available across each platform supported by SQL Anywhere. Note that, with a few exceptions, components that are available in all supported platforms are not listed. For ease of reading, the platforms are grouped by operating system vendor, operating system name and processor architecture.
SQL Anywhere Supported Linux Platforms	http://www.sybase.com/detail?id=1035824	Provides information about the Linux platform support, including support exceptions, key components, and tested Linux distributions.
SQL Anywhere Supported Kerberos Clients	http://www.sybase.com/detail?id=1061807	Provides a list of Kerberos clients/runtimes that have been tested with SQL Anywhere.
SQL Anywhere Supported Listener Platforms	http://www.sybase.com/detail?id=1061808	Provides a list of supported Listener platforms for SQL Anywhere.
ODBC Drivers for MobiLink	http://www.sybase.com/detail?id=1011880	Provides a list of the recommended ODBC drivers for different versions of the MobiLink server.

Web page name	URL	Description
SQL Anywhere Client Interfaces	http://www.sybase.com/detail?id=1068981	Provides a list of supported client interfaces for SQL Anywhere.

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

Accessibility Enablement option

SQL Anywhere 12 includes an accessibility enablement module. This component provides the Java Access Bridge module, which is loaded whenever you use Sybase Central or Interactive SQL. Third-party software such as screen readers use 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=1061806.

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

Data management technologies

SQL Anywhere offers two relational 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, full-text searching, support for spatial data, 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 iPhone and BlackBerry. It provides full transaction-processing, a choice of development models, and a built-in synchronization client for exchanging data with other databases.

UltraLiteJ is a Java implementation of a UltraLite, which is designed for BlackBerry smartphones and for Java SE. See "Introduction to UltraLiteJ" [*UltraLiteJ*].

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** A database is 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.

  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, 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.

  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 16.

- **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 23.

  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.

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

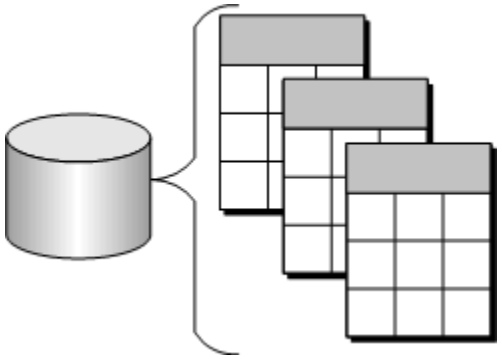
  If you develop an application using a rapid application development (RAD) tool such as Sybase PowerBuilder, you may find that the tool provides its own methods for communicating with database servers, and hides the details of the language interface. Nevertheless, all applications use one of the supported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

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 Microsoft Visual FoxPro. This document uses the informal terms.

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

Informal relational term	Formal relational term	Non-relational term
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 a SQL Anywhere database" [*SQL Anywhere Server - Database Administration*].

Relationships 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 "How to manage text indexes" [*SQL Anywhere Server - SQL Usage*], and "Full text search" [*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" [*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
- Spatial Reference systems

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 SQL Anywhere database server

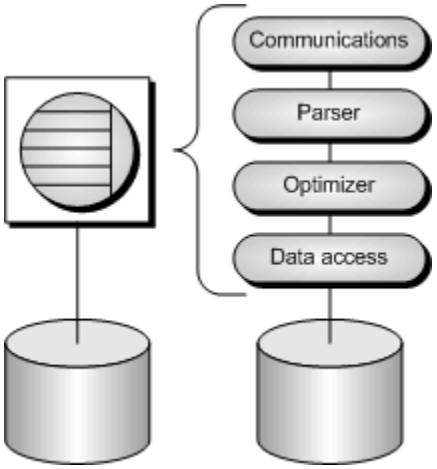
The SQL Anywhere database server has an internal structure that allows many requests to be handled efficiently.

- 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 *dbeng12.exe*. (It is not provided on Windows Mobile.)
  - On Unix operating systems, it is *dbeng12*.

- 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 *dbsrv12.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 *dbsrv12*.

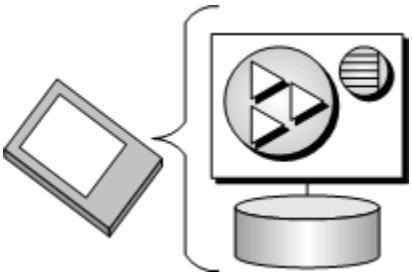
If you have received SQL Anywhere as part of another product, you may not have both versions of the database server. Similarly, not all components are available on all operating systems. For example, there is no personal server for Windows Mobile, only a network server.

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

Inside UltraLite

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. Differences 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, TMPDIR, and TEMP environment variables" [*SQL Anywhere Server - Database Administration*].

UltraLite database files

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 BlackBerry, Embedded Linux, or iPhone, you must use an UltraLite database.
- If the target platform is Windows 7, 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.

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

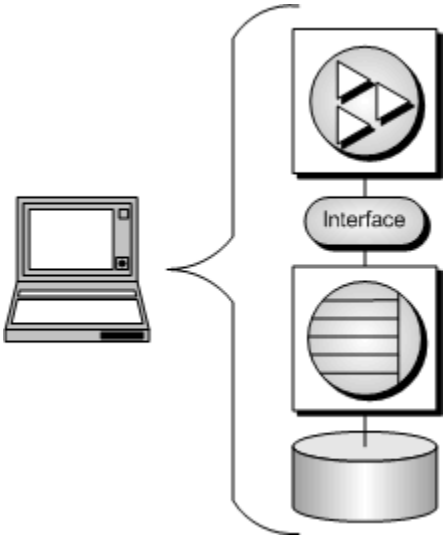
Database applications can connect to a database server located on the same computer as the application. Network database servers can connect to a different computer. In addition, with SQL Anywhere you can build distributed databases for remote office and mobile applications, with physically distinct 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.

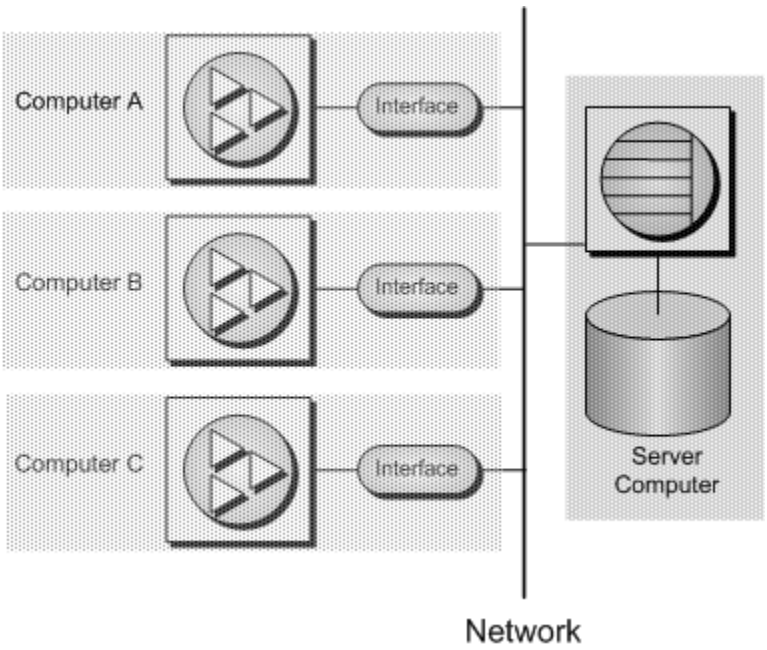
Many relational database management systems require experienced staff for administration. A characteristic of SQL Anywhere databases is the ability to run entirely without administration.

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

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 over TCP/IP.

For a client application to work in a client/server environment, you need to specify additional connection parameters, typically the HOST connection parameter and optionally the ServerName parameter.

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 Sybase PowerBuilder components, Java beans, or COM components.

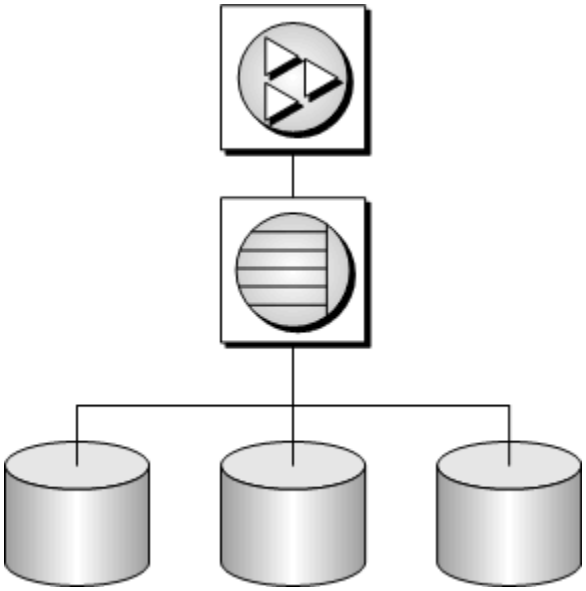
Application servers can also provide transaction logic to their client applications—guaranteeing that sets of operations are executed atomically across multiple databases. SQL Anywhere is well suited to multi-tier computing, and can participate in distributed transactions coordinated by Microsoft Distributed 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 (DBF) connection parameter" [*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*].

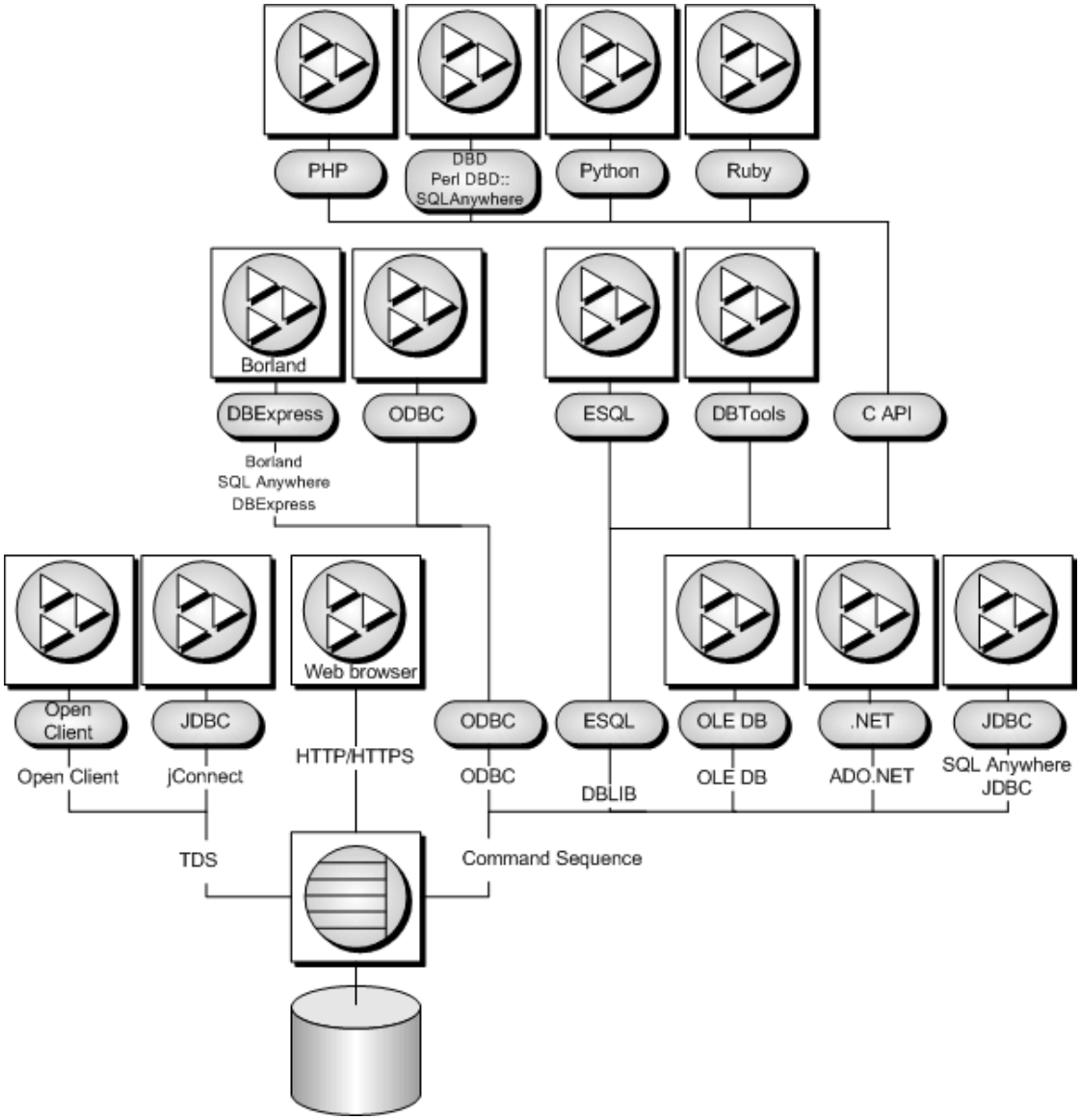
Programming interfaces

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 19.

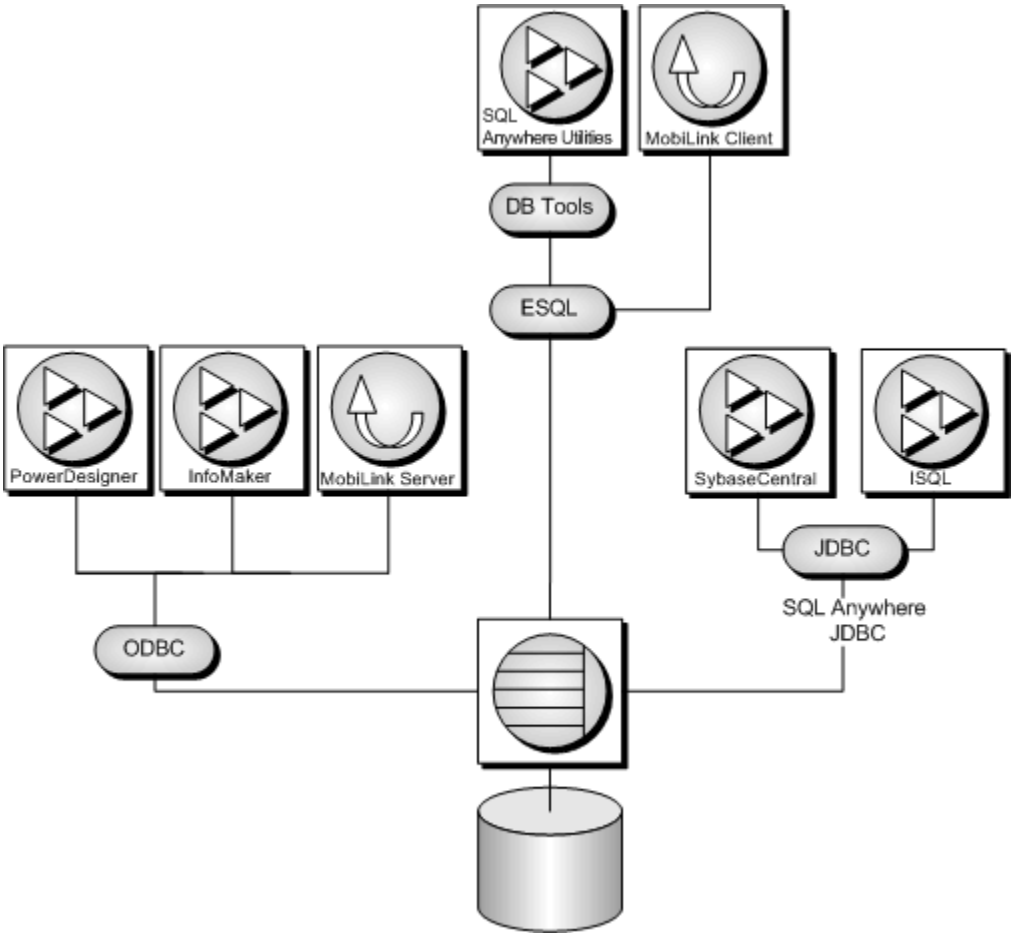
Supported SQL Anywhere programming interfaces and protocols

The following diagram displays the supported interfaces, and the interface libraries used. The name of the interface library and the interface are usually the same.



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 .NET support" [*SQL Anywhere Server - Programming*]
- "ODBC support" [*SQL Anywhere Server - Programming*]
- "OLE DB and ADO development" [*SQL Anywhere Server - Programming*]
- "Embedded SQL" [*SQL Anywhere Server - Programming*]
- "JDBC support" [*SQL Anywhere Server - Programming*]
- "Sybase Open Client support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere C API support" [*SQL Anywhere Server - Programming*]
- "Perl DBI support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere PHP extension" [*SQL Anywhere Server - Programming*]
- "Python support" [*SQL Anywhere Server - Programming*]
- "SQL Anywhere Ruby API support" [*SQL Anywhere Server - Programming*]

UltraLite programming interfaces

UltraLite also provides you with 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 parameters are different.

- **Command Sequence** This protocol is used by SQL Anywhere, the SQL Anywhere JDBC driver, and the embedded SQL, ODBC, OLE DB, and ADO.NET APIs.
- **TDS** This protocol is used by Sybase Adaptive Server Enterprise, the jConnect JDBC driver, and Sybase Open Client applications.

Overview of 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 "Introducing SQL Remote" [*SQL Remote*].

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.

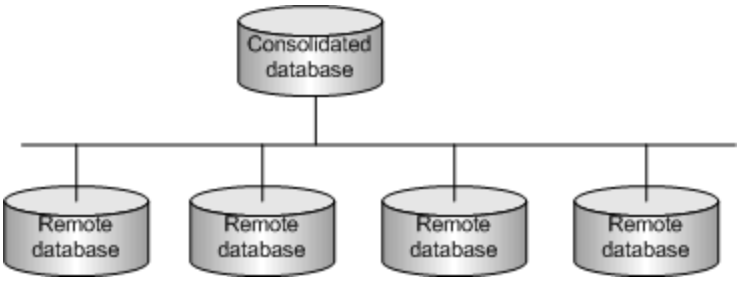
In MobiLink, the **consolidated database** is one of several supported RDBMSs. The consolidated database, which typically resides on a corporate server, tracks synchronization information and optionally contains the data to be replicated. Other central data may be stored in any other format, such as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

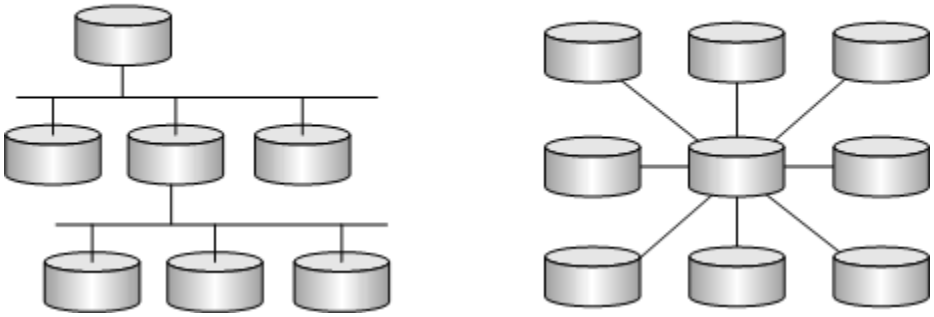
Hierarchical database configurations

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.

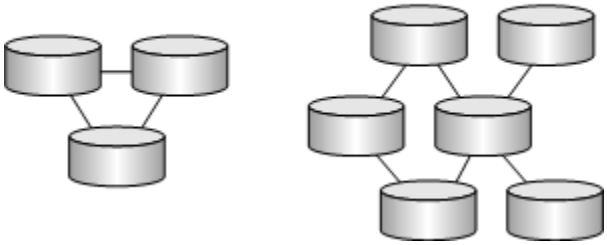
For any two databases directly sharing data in a hierarchical configuration, one is always above or below the other in the hierarchy.

Hierarchical database configurations



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

Non-hierarchical configurations



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.

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.

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.

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.

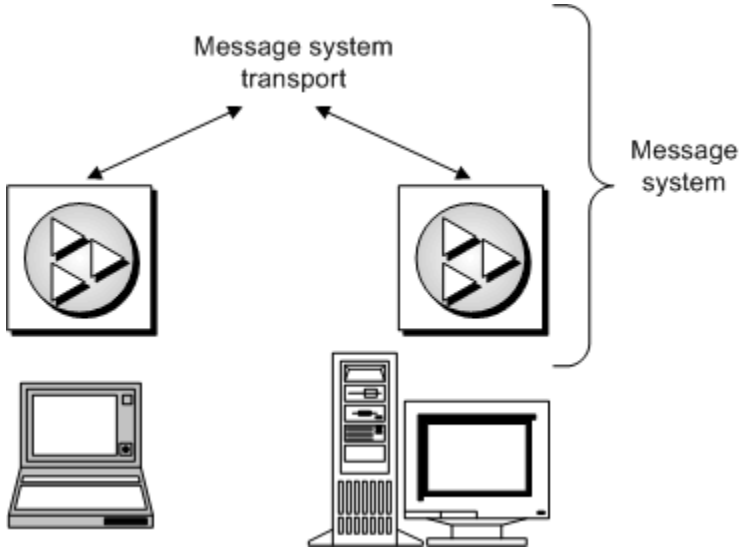
In message-based communications, each message carries its destination address and other control information so that no direct connection is necessary between applications exchanging information. For example, an email message contains the destination address; there is no direct connection between the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

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 the 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 before 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 38.

  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:

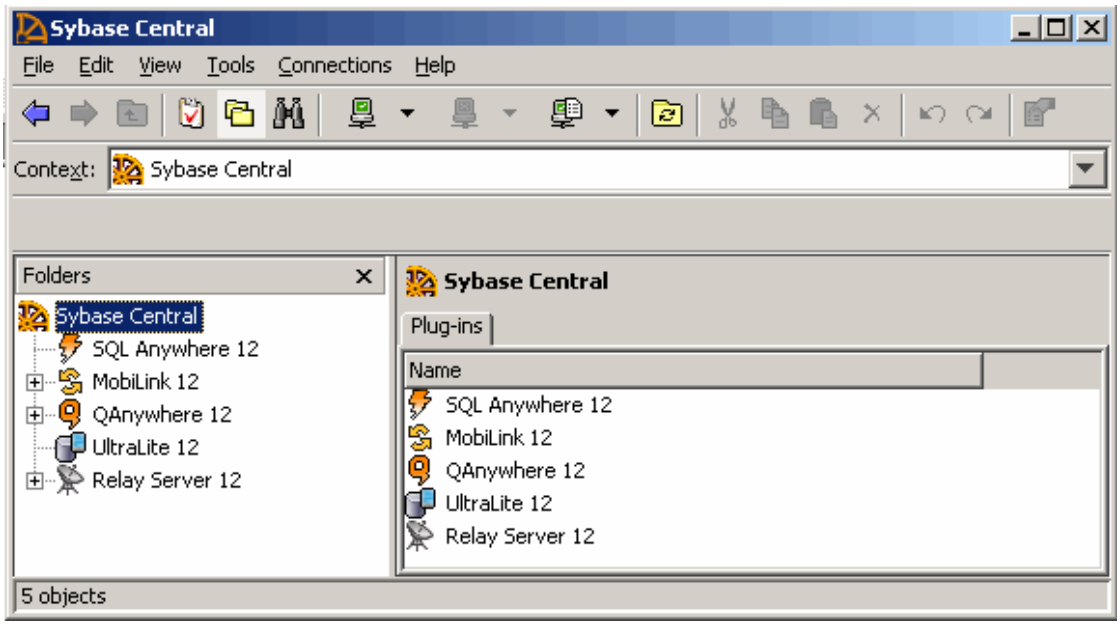
- You want to access a web service from a mobile application.
- A network connection between the various systems is not continually available.

See "Mobile web services" [*QAnywhere*].

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, Relay Server, and other Sybase products.



In addition to helping with routine tasks, Sybase Central also provides performance statistics, procedure profiling, 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:

  - **Support for spatial data** - Spatial data is data that describes the position, shape, and orientation of objects in a defined space. Spatial data in SQL Anywhere is represented as 2D geometries in the form of points, curves (line strings and strings of circular arcs), and polygons. See "SQL Anywhere Server - Spatial Data Support".

  - **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, MobiLink servers, MobiLink server farms, and Relay Server farms. The Monitor provides constant data collection, email alert notifications, a browser-based interface, and the ability to monitor multiple databases, MobiLink servers, MobiLink server farms, and Relay Server Farms. See "SQL Anywhere Monitor" [*SQL Anywhere Server - Database Administration*], "SQL Anywhere Monitor for MobiLink" [*MobiLink - Server Administration*], and "SQL Anywhere Monitor for Relay Server" [*Relay Server*].

- **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*].

- **Relay Server** The Relay Server enables secure, load-balanced communication between mobile devices and back-end servers through a web server. Supported back-end servers include MobiLink, Unwired Server, Afaria, and Mobile Office. See "Relay Server"

- **Utilities** SQL Anywhere includes various utilities for performing administration tasks such as backing up a database and performing synchronizations. Utilities are useful for including in batch files for repeated use. See:

  - "Database administration utilities" [*SQL Anywhere Server - Database Administration*]
  - "UltraLite utilities" [*UltraLite - Database Management and Reference*]
  - "MobiLink utilities" [*MobiLink - Server Administration*]
  - "QAnywhere Agent utilities reference" [*QAnywhere*]
  - "SQL Remote utilities and options reference" [*SQL Remote*]

- **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 http://www.sybase.com/products/development/infomaker.

- **PowerDesigner Physical Data Model** SQL Anywhere includes Physical Data Model, a module of the powerful database design tool, Sybase 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 Sybase PowerDesigner, see http://www.sybase.com/products/modelingmetadata/powerdesigner.

  For information about SQL Anywhere database design, see "Creating a SQL Anywhere database" [*SQL Anywhere Server - Database Administration*].

Sample databases

This section describes the schemas of the SQL Anywhere 12 sample databases. Experiment with the sample databases to learn more about SQL Anywhere 12.

SQL Anywhere 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)

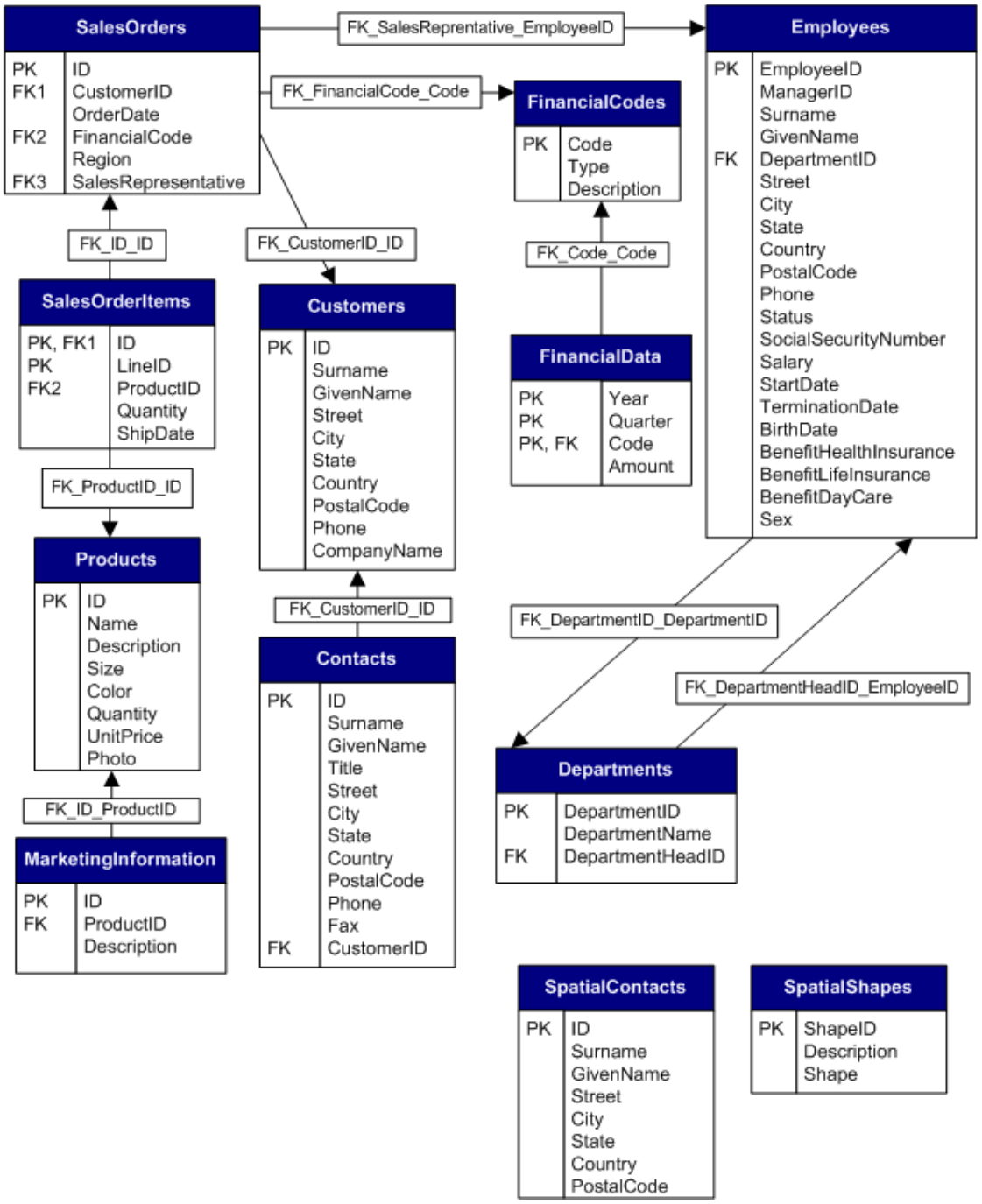
Caution
The sample database always has the same user ID and password; it is recommended that you change the DBA user ID and password to restrict access to the database. See "Changing a password" [*SQL Anywhere Server - Database Administration*].

The sample database uses the following ODBC data source: **SQL Anywhere 12 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*].



Recreate the sample database (demo.db)

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 (Windows)

1. Run the following command to erase *demo.db* and create a new copy of the sample database with objects and data:

   ```
   newdemo %SQLANYSAMP12%\demo.db
   ```

2. When you are prompted, choose to erase any existing files.

To recreate the sample database (Unix)

1. Run the following command in the directory where the sample database is located to erase *demo.db* and create a new copy of the sample database with objects and data:

   ```
   newdemo.sh demo.db
   ```

2. When you are prompted, choose to erase any existing files.

To create a copy of the sample database with a different name (Windows)

- Run the following command to create a database called *mydemo.db* that contains objects and data.

  If you do not specify a path, the file is created in the current directory.

  ```
  newdemo path\mydemo.db
  ```

To create a copy of the sample database with a different name (Unix)

- Run the following command to create a database called *mydemo.db* that contains objects and data.

  If you do not specify a path, the file is created in the current directory.

  ```
  newdemo.sh path/mydemo.db
  ```

The CustDB sample database application

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*].

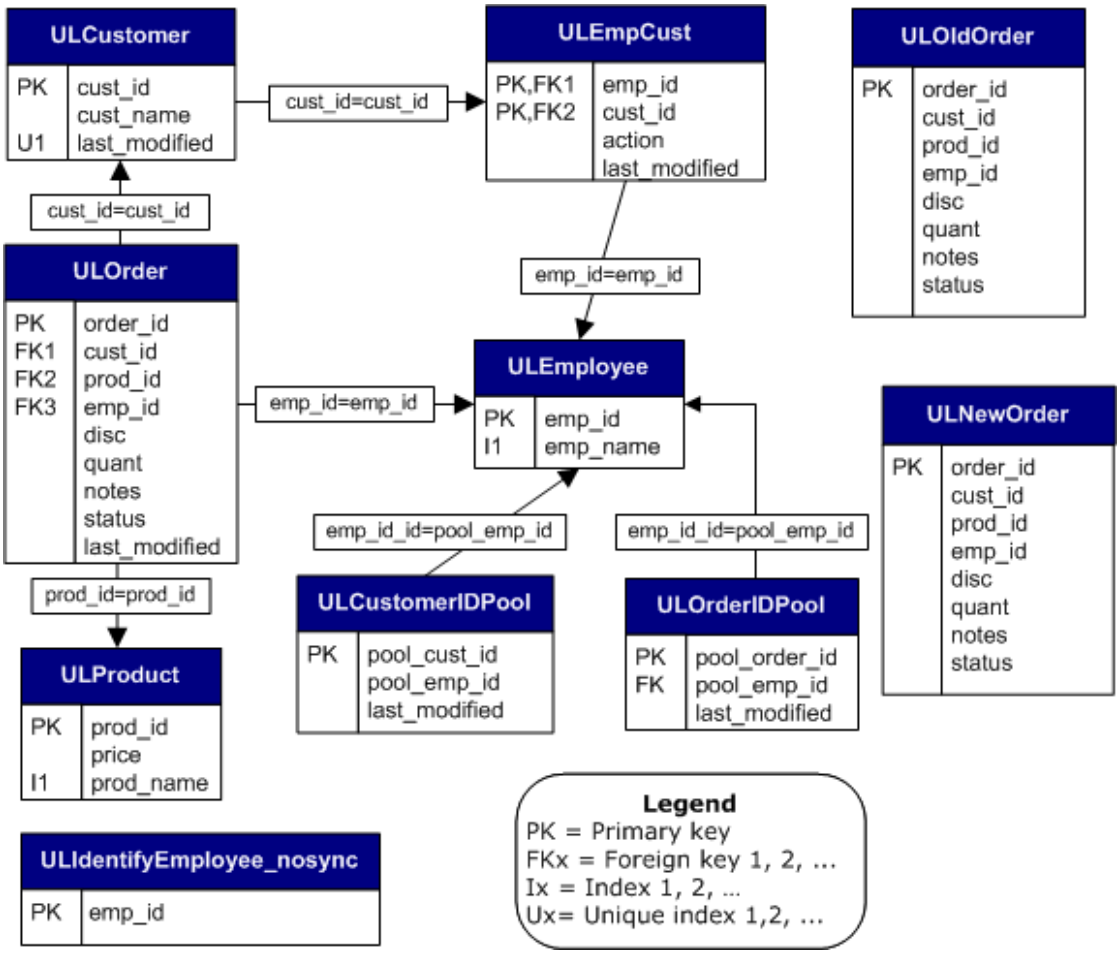
- **MobiLink** If you are interested in exploring MobiLink features, there is a CustDB consolidated database that contains sample synchronization logic. You can use this consolidated database with the CustDB UltraLite sample remote database to run the sample synchronization system.

  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 IBM DB2). These setup files are located in *samples-dir**MobiLink**CustDB*\.

  The CustDB consolidated database uses the following ODBC data source: **SQL Anywhere 12 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
ULOldOrder
PK order_id
cust_id
prod_id
emp_id
disc
quant
notes
status
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
ULNewOrder
PK order_id
cust_id
prod_id
emp_id
disc
quant
notes
status
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
Legend
PK = Primary key
FKx = Foreign key 1, 2, ...
Ix = Index 1, 2, ...
Ux= Unique index 1,2, ...
ULIdentifyEmployee_nosync
PK emp_id

Getting started with SQL Anywhere 12

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

For information about *demo.db*, see "SQL Anywhere sample database" on page 43.

For information about *custdb.db*, see:

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

Getting started resources

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 50.

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

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

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

- **Relay Server** See "Relay Server".

  This book explains how to set up and use the Relay Server.

Other applications

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

- **PowerDesigner Physical Data Model** See "PowerDesigner Physical Data Model" on page 41.

  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 "InfoMaker" on page 41.

  You can use InfoMaker to create reports quickly.

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. 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 39.

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

  Or, run the following command:

  ```
  dbeng12 -?
  ```

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

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

  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

Frequently asked questions - SQL Anywhere

What is the default user ID and password for demo.db and newly created databases?

The default user ID for demo.db and newly created databases is **DBA** and the default password is **sql**. The user ID is case-insensitive and the password is case-sensitive. You should change the password before deploying the database.

For more information about default user IDs and passwords, see "DBA authority" [*SQL Anywhere Server - Database Administration*].

How do I recover a DBA password?

A lost or forgotten DBA password cannot be recovered. If you require assistance, contact Technical Support (http://www.sybase.com/support).

How do I connect to a database?

The connection parameters required to connect to a database vary depending on the location of the application relative to the database server. The following connection scenarios assume that there is no communication encryption and there is only one database running on a database server.

In the following examples, you connect to the SQL Anywhere sample database, *demo.db* using Interactive SQL. This file is installed in the SQL Anywhere samples directory represented as *samples-dir* in the following example. For information about the default location of *samples-dir*, see "Samples directory" [*SQL Anywhere Server - Database Administration*].

- **Example 1: The database server is not running, and the database file is installed on the same computer as the application** You must provide the correct user ID, password, connection and database file parameters for the database file. You may also need to specify the database server name. To connect to the database, run a command similar to the following:

  ```
  dbisql -c "UID=DBA;PWD=sql;Server=demo;DBF=samples-dir\demo.db"
  ```

- **Example 2: The application and the database server are running on the same computer** You must provide the correct user ID and password. It is also recommended that you provide the database server name to avoid connecting to the wrong database server. To connect to the database, run a command similar to the following:

  ```
  dbisql -c "UID=DBA;PWD=sql;Server=demo"
  ```

- **Example 3: The application and database server are running on different computers** You must provide the correct user ID, password, database server name, and HOST connection parameters. To connect to the database, run a command similar to the following:

  ```
  dbisql -c "UID=DBA;PWD=sql;Server=demo;HOST=myhost"
  ```

How do I connect to a database on Unix?

See "Connect to the sample database on Unix" [*SQL Anywhere Server - Database Administration*].

How do I connect to a database on Mac OS X?

See "Connect to the sample database on Mac OS X" [*SQL Anywhere Server - Database Administration*].

How do I determine which version of SQL Anywhere was used to create a database?

Databases created with SQL Anywhere 9.0.1 or later include the SYSHISTORY system view. The rows in this view that reflect INIT and UPDATE operations give information about when, and with what version, the database was created and upgraded. See "SYSHISTORY system view" [*SQL Anywhere Server - SQL Reference*].

To determine the version without starting the database, see "DBCreatedVersion function" [*SQL Anywhere Server - Programming*].

Can I install and run two different versions of SQL Anywhere on the same computer?

Yes, you can install multiple major versions of SQL Anywhere on the same computer. For example, SQL Anywhere 9.0.2, 10.0.1, and 11.0.1 can all be installed and run independently.

However, you must use caution when starting a SQL Anywhere executable that has the same name in multiple versions (for example, dbisql or dbinit) to ensure that you are starting the correct version of the application. You can either specify the full absolute path using environment variables such as SQLANY12, or ensure the version of SQL Anywhere you want is specified first in your path.

See "Using the utilities" [*SQL Anywhere 12 - Changes and Upgrading*].

Can I install and run two copies of the same major version of SQL Anywhere on the same computer?

For SQL Anywhere 10 and earlier for Windows using the SQL Anywhere installer Yes, you can install multiple copies of the same version of SQL Anywhere on the same computer. However, the SQL Anywhere installation program registers some drivers and components in the Windows registry, and there is only one copy of the registry. As a result, the ODBC and OLE DB drivers from the most recent installation are used. Similarly, there is only one set of **Start** menu shortcuts. They point to the most recently installed copy of SQL Anywhere.

For SQL Anywhere 11 and later for Windows using the SQL Anywhere installer No, you can only install a single copy of SQL Anywhere 11 on a computer.

For SQL Anywhere 12 and earlier for Linux or Unix using the SQL Anywhere installer Yes, you can install multiple versions in different locations. However, on Linux, if you choose to install application menu items, each user can only have one set of application menu items. They point to the most recently installed copy of SQL Anywhere.

For deployed embedded database applications Yes, deployed embedded database applications that include SQL Anywhere can be deployed with other SQL Anywhere installations on the same computer.

On Windows operating systems, the ODBC and OLE DB driver names in the registry should include the name of the application in which they are embedded. For example, the ODBC driver name **SQL Anywhere 12** should be renamed *application-name* **SQL Anywhere 12**. See "Deploying databases and applications" [*SQL Anywhere Server - Programming*] and "Configuring the ODBC driver" [*SQL Anywhere Server - Programming*].

Is it better to have one database server for each database or should I run multiple databases on a single database server?

You should run multiple databases on a single database server because this configuration optimizes the use of computer resources.

Having multiple database servers running on the same computer may result in a competition for resources, and (with dynamic cache resizing) this configuration may result in degraded or unpredictable performance. Degraded or unpredictable performance may be acceptable if you need to stop one database server for maintenance without affecting the others, or if you need to isolate errors to a single database server.

You should verify that you are correctly licensed for the installation option that you choose. For more information, see "Running multiple databases on a single database server" on page 21.

Why is the size of my database increasing or not decreasing as expected?

See "Understanding unexpected changes in the database size" [*SQL Anywhere Server - Database Administration*].

How do I create an effective backup and recovery plan?

See the following:

- "Designing a backup and recovery plan" [*SQL Anywhere Server - Database Administration*]
- See the tech note titled What Backup, Recovery, and Disaster Recovery Mean to Your SQL Anywhere Databases (http://www.sybase.com/detail?id=47877)

What do I do when an assertion failure occurs?

See the tech note titled I've Got An Assertion! What Should I do? (http://www.sybase.com/detail?id=1010805).

How do I report a bug?

To report low priority SQL Anywhere bugs, log in to http://case-express.sybase.com. Bugs reported using Case-Express are assigned a lower priority than cases opened through Technical Support.

For priority issues, open a support case with Technical Support (http://www.sybase.com/support).

How do I improve the performance of my application or database server?

See the following documents:

- "Improving database performance" [*SQL Anywhere Server - SQL Usage*]
- "Performance improvement tips" [*SQL Anywhere Server - SQL Usage*]
- Diagnosing Application Performance Issues with SQL Anywhere (http://www.sybase.com/detail?id=1060302)
- Capacity Planning with SQL Anywhere (http://www.sybase.com/detail?id=1056535)

How do I upgrade my SQL Anywhere software?

See "Upgrading SQL Anywhere" [*SQL Anywhere 12 - Changes and Upgrading*].

Why is my application running slower after an upgrade?

SQL Anywhere performs best when the database is created with the same major version as the database server. Rebuild the database if you are experiencing performance problems and no longer need to run the database on earlier versions of SQL Anywhere. See "Rebuilding databases" [*SQL Anywhere Server - SQL Usage*] and "Improving database performance" [*SQL Anywhere Server - SQL Usage*].

Why does my application not work after an upgrade?

SQL Anywhere developers strive to ensure that applications continue to work after an upgrade. However, your application may be affected by behavior changes and the removal of previously supported features.

To determine if behavior changes have been made to your version of SQL Anywhere, or if features have been removed or deprecated, see "SQL Anywhere 12 - Changes and Upgrading". Select the section relevant to your version of SQL Anywhere.

What operating systems are supported by the different versions of SQL Anywhere?

See the tech note SQL Anywhere Supported Platforms and Engineering Support Status (http://www.sybase.com/detail?id=1002288).

What are the licensing requirements for a typical SQL Anywhere installation?

For descriptions of the different licensing options for SQL Anywhere 10 and later, including examples that show you how to apply your license, see SQL Anywhere Licensing (http://www.sybase.com/detail?id=1056242).

What is the largest database size supported by SQL Anywhere?

The size of the database that can be supported by SQL Anywhere is dependent on the memory, CPU, and disk drive capacity of the computer on which SQL Anywhere is installed. See "SQL Anywhere size and number limitations" [*SQL Anywhere Server - Database Administration*].

A number of our customers discuss the implementation of large databases in the newsgroups. See the newsgroup thread (http://forums.sybase.com/cgi-bin/webnews.cgi?cmd=item-99571&group=sybase.public.sqlanywhere.general).

How do I migrate from another database product to SQL Anywhere?

To migrate to SQL Anywhere, you must import your data into a SQL Anywhere database. See "Migrating databases to SQL Anywhere" [*SQL Anywhere Server - SQL Usage*].

There are differences between products, such as different dialects of SQL, so your application may need to be modified.

What competitive advantages does SQL Anywhere offer?

See Choosing SQL Anywhere for ISV Applications (http://www.sybase.com/detail?id=1053363).

I need to diagnose performance issues with a specific query. How do I create a graphical plan with statistics?

See "To create a graphical plan with detailed and node statistics" [*SQL Anywhere Server - Database Administration*].

Can I use the OUTPUT statement in a stored procedure?

No. The OUTPUT statement can only be executed in Interactive SQL and cannot be used within a stored procedure. Use the UNLOAD statement within your stored procedure to save the result set generated by a SQL statement to a text file. If you use the UNLOAD statement in your stored procedure, information is unloaded from the database server and not from the client computer from which the command was executed. See "UNLOAD statement" [*SQL Anywhere Server - SQL Reference*].

Where can I find more information?

See:

- SQL Anywhere documentation: dcx.sybase.com/dcx_home.html
- SQL Anywhere v10 / v9.0.2 Frequently Asked Questions (http://www.sybase.com/detail?id=1053692)
- SQL Anywhere 11.0.1 FAQs (http://www.sybase.com/detail?id=1062382)
- SQL Anywhere Studio/Adaptive Server Anywhere Patches and Upgrades FAQ (http://www.sybase.com/detail?id=1019567)
- SQL Anywhere Web Edition FAQ (http://www.sybase.com/detail?id=1057560)
- SQL Anywhere newsgroups (http://www.sybase.com/detail?id=1002557)
- SQL Anywhere Product Page (http://www.sybase.com/products/databasemanagement/sqlanywhere)

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

