SYBASE iAnywhere

SQL Anywhere® 12
Changes and Upgrading

Contents

About this book

This book describes new features in SQL Anywhere 12 and in previous versions of the software.

For information about new features and behavior changes in versions of SQL Anywhere before version 8, go to http://dcx.sybase.com/html/dbwnen10/dbwnen10.html.

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" [*SQL Anywhere 12 - Introduction*].

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.

What's new in version 12.0.0

For information about new features and behavior changes in versions of SQL Anywhere before version 9, see http://dcx.sybase.com/html/dbwnen10/dbwnen10.html.

Product-wide new features

Following is a list of product-wide additions introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **Escape characters supported for configuration files** The parsing for configuration files has been enhanced to support \\ as an escape sequence for a \, and \" as an escape sequence for a ". See "Configuration file escape characters" [*SQL Anywhere Server - Database Administration*].

Product-wide behavior changes

Following is a list of product-wide behavior changes introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **SQL Anywhere Replication Agent for Sybase Replication Server unsupported** The SQL Anywhere Replication Agent for Sybase Replication Server is not supported in version 12. You must use an alternative replication or synchronization technology such as MobiLink or SQL Remote. See "Introducing MobiLink Technology" [*MobiLink - Getting Started*] and "Introducing SQL Remote" [*SQL Remote*].

  The following changes have been made to the software as a result of this change:

  - **a_change_log DBTools structure** The ignore_ltm_trunc member is no longer supported.
  - **LTMGeneration database property** This property is reserved for system use.
  - **LTMTrunc database property** This property is reserved for system use.
  - **Log Transfer Manager utility (dbltm)** This utility has been removed.
  - **Log Translation utility (dbtran)** The -is option no longer supports the value RepServer.

    The -rsu option has been removed.
  - **Service utility (dbsvc)** You can no longer create services for Replication Agent. The SQLANYLTM service group is no longer supported.

    The -w and -t options no longer support the value **dbltm**.
  - **Support utility (dbsupport)** This utility no longer returns information for the SQL Anywhere Replication Agent (dbltm).

- **Transaction Log utility (dblog)** The -g and -il options are no longer supported. See "Transaction Log utility (dblog)" [*SQL Anywhere Server - Database Administration*].
- **replicate_all database option** This option has been removed.
- **delete_old_logs database option** This option is not supported for use with Replication Agent.
- **ALTER PROCEDURE statement** The following syntax is no longer supported:

  ALTER PROCEDURE [*owner*.]*procedure-name*
  REPLICATE { **ON** | **OFF** }

  See "ALTER PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*].
- **ALTER TABLE statement** The **REPLICATE** { **ON** | **OFF** } clause is no longer supported. See "ALTER TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **Executables now respect the user's umask settings when running as a daemon** In previous releases when an executable ran as a daemon (it was started with the -ud option) on Unix, the executable ignored the user's umask setting and called umask(0), which created new files with group +other read/write permissions. When you start a SQL Anywhere 12 executable as a daemon, the executable does not call umask(0) and respects the user's umask setting. Because the current user's umask setting controls the permissions for executables, you should ensure that the user's umask value is set to the desired level before starting the executable.

  This behavior change applies to the following executables:

  - dbeng12
  - dbsrv12
  - dbltm
  - dbmlsync
  - dbns12
  - dbremote
  - mlsrv12
  - uleng12

SQL Anywhere new features

Following is a list of new features in SQL Anywhere version 12.0.0.

For information about changes to the list of supported platforms, see http://www.sybase.com/detail?id=1061806.

Main features

Following is a list of the main features introduced in SQL Anywhere version 12.0.0.

- **New spatial data support** The following features have been added in support of the new spatial data capabilities in SQL Anywhere 12.0.0. You must upgrade your database to use this feature.

  Note
  Spatial data support for 32-bit Windows and 32-bit Linux requires a CPU that supports SSE2 instructions. This support is available with Intel Pentium 4 or later (released in 2001) and AMD Opteron or later (released in 2003).

  SQL statements The following SQL statement enhancements have been made in support of the spatial feature:

  - **SHAPEFILE clause** A new SHAPEFILE format option is available for the OPENSTRING subclause of the FROM clause. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

    As well, a new SHAPEFILE format option is available for the FORMAT clause of the LOAD TABLE statement. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

  - **CREATE SPATIAL REFERENCE SYSTEM statement** Creates or replaces a spatial reference system. See "CREATE SPATIAL REFERENCE SYSTEM statement" [*SQL Anywhere Server - SQL Reference*].

  - **ALTER SPATIAL REFERENCE SYSTEM statement** Changes the settings of an existing spatial reference system. See the Remarks section for considerations before altering a spatial reference system. See "ALTER SPATIAL REFERENCE SYSTEM statement" [*SQL Anywhere Server - SQL Reference*].

  - **DROP SPATIAL REFERENCE SYSTEM statement** Drops a spatial reference system. See "DROP SPATIAL REFERENCE SYSTEM statement" [*SQL Anywhere Server - SQL Reference*].

  - **CREATE SPATIAL UNIT OF MEASURE statement** Creates or replaces a spatial unit of measurement. See "CREATE SPATIAL UNIT OF MEASURE statement" [*SQL Anywhere Server - SQL Reference*].

  - **DROP SPATIAL UNIT OF MEASURE statement** Drops a spatial unit of measurement. See "DROP SPATIAL UNIT OF MEASURE statement" [*SQL Anywhere Server - SQL Reference*].

  Interactive SQL changes A new viewer tool, the **Spatial Viewer**, has been added to Interactive SQL to allow you to view spatial geometries. You can query spatial data in the top portion of the viewer, and then see your results represented as an image in the lower portion of the viewer. See "View spatial data as images" [*SQL Anywhere Server - Spatial Data Support*].

  Also, when viewing a result row in Interactive SQL, you can now preview a geometry as a Scalable Vector Graphic (SVG) using the new **Spatial Preview** tab. See "View spatial data as images" [*SQL Anywhere Server - Spatial Data Support*].

  New data types, methods, and constructors New types, methods, and constructors have been added to allow you access, model, and analyze spatial data. See "Accessing and manipulating spatial data" [*SQL Anywhere Server - Spatial Data Support*].

As well, many spatial compatibility functions have been provided to mimic regular SQL functions when accessing and manipulating spatial data. These functions have been provided for compatibility with other products, and make use of the spatial methods and constructors provided in SQL Anywhere. See "Spatial compatibility functions" [*SQL Anywhere Server - Spatial Data Support*].

New functions and system procedures The following functions and system procedures have been added in support of spatial data in the database:

- **TREAT function** Allows you to change the declared type of a geometry expression to a subtype. See "TREAT function [Data type conversion]" [*SQL Anywhere Server - SQL Reference*].
- **sa_describe_shapefile system procedure** Describes the names and types of columns contained in a ESRI shapefile. This system feature is for use with the spatial feature. See "sa_describe_shapefile system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_install_feature system procedure** Installs additional features that were not present in the database when SQL Anywhere was installed. See "sa_install_feature system procedure" [*SQL Anywhere Server - SQL Reference*].
- **st_geometry_dump system procedure** Expands a geometry object into a result set with each row representing one of the geometry objects contained in the input. See "st_geometry_dump system procedure" [*SQL Anywhere Server - SQL Reference*].

Wizards In Sybase Central, the following wizards have been added in support of the spatial data feature:

- **Create Spatial Reference System wizard** The **Create Spatial Reference System** wizard allows you to create new spatial reference systems. See "Create a spatial reference system" [*SQL Anywhere Server - Spatial Data Support*].
- **Create Unit Of Measure wizard** The **Create Unit Of Measure** wizard allows you to create new units of measure for use with spatial data. See "Create a unit of measure" [*SQL Anywhere Server - Spatial Data Support*].

Catalog changes The following changes have been made to the catalog as part of the new spatial data support:

- **SYSSPATIALREFERENCESYSTEM system view** Each row of the SYSSPATIALREFERENCESYSTEM system view describes a spatial reference system defined in the database. See "SYSSPATIALREFERENCESYSTEM system view" [*SQL Anywhere Server - SQL Reference*].
- **SYSUNITOFMEASURE system view** Each row of the SYSUNITOFMEASURE system view describes a unit of measure defined in the database. See "SYSUNITOFMEASURE system view" [*SQL Anywhere Server - SQL Reference*].
- **ST_GEOMETRY_COLUMNS consolidated view** Each row of the ST_GEOMETRY_COLUMNS system view describes a spatial column defined in the database. See "ST_GEOMETRY_COLUMNS consolidated view" [*SQL Anywhere Server - SQL Reference*].

- **ST_SPATIAL_REFERENCE_SYSTEMS consolidated view** Each row of the ST_SPATIAL_REFERENCE_SYSTEMS system view describes a spatial reference system defined in the database. See "ST_SPATIAL_REFERENCE_SYSTEMS consolidated view" [*SQL Anywhere Server - SQL Reference*].

- **ST_UNITS_OF_MEASURE consolidated view** Each row of the ST_UNITS_OF_MEASURE system view describes a unit of measure defined in the database. See "ST_UNITS_OF_MEASURE consolidated view" [*SQL Anywhere Server - SQL Reference*].

Database options and properties The following database options and properties have been added in support of the spatial data features.

- **st_geometry_asbinary_format option** Controls how spatial values are converted from a geometry to binary. See "st_geometry_asbinary_format option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_astext_format option** Controls how spatial values are converted from a geometry to text. See "st_geometry_astext_format option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_asxml_format option** Controls how spatial values are converted from a geometry to XML. See "st_geometry_asxml_format option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_describe_type option** Controls how spatial values are described. See "st_geometry_describe_type option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_on_invalid option** Controls the behavior when a geometry fails basic validation. See "st_geometry_on_invalid option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_asbinary_format connection property** Returns a value that indicates how spatial values are converted from a geometry to binary. See "st_geometry_asbinary_format connection property" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_astext_format connection property** Returns a value that indicates how spatial values are converted from a geometry to text. See "st_geometry_astext_format connection property" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_asxml_format connection property** Returns a value that indicates how spatial values are converted from a geometry to xml. See "st_geometry_asxml_format connection property" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_describe_type connection property** Returns a value that indicates how spatial values are described to the client. See "st_geometry_describe_type connection property" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_on_invalid connection property** Returns a value that indicates the behavior when a geometry fails basic validation. See "st_geometry_on_invalid connection property" [*SQL Anywhere Server - Database Administration*].

SYS_SPATIAL_ADMIN_ROLE group Membership in this group allows users to create, alter, or drop spatial reference systems and units of measure. See "Spatial permissions" [*SQL Anywhere Server - Spatial Data Support*].

For more information about SQL Anywhere spatial support, see "Getting started with spatial data" [*SQL Anywhere Server - Spatial Data Support*].

- **Read-only scale-out** You can now use SQL Anywhere in a read-only scale-out system. In this configuration, one database server (the root node) runs a read-write copy of the database, while other database servers run read-only copies of the database (copy nodes) that can be used to offload reporting and other operations that require read access to the database. Read-only scale-out can be used on its own, or with database mirroring. You must upgrade or rebuild existing databases to use this feature.

  A sample has been added in *samples-dir\SQLAnywhere\DBMirror* that uses a database mirroring system in conjunction with a scale-out system.

  See:

  - "SQL Anywhere read-only scale-out" [*SQL Anywhere Server - Database Administration*]
  - "NodeType (NODE) connection parameter" [*SQL Anywhere Server - Database Administration*]
  - "CREATE MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
  - "SET MIRROR OPTION statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
  - MIRROR SERVER clause: "COMMENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "SYSMIRROROPTION system view" [*SQL Anywhere Server - SQL Reference*]
  - "SYSMIRRORSERVER system view" [*SQL Anywhere Server - SQL Reference*]
  - "SYSMIRRORSERVEROPTION system view" [*SQL Anywhere Server - SQL Reference*]
  - "MirrorRole database property" [*SQL Anywhere Server - Database Administration*]
  - MirrorServerState and MirrorState properties: "sa_mirror_server_status system procedure" [*SQL Anywhere Server - SQL Reference*]

- **Database mirroring enhancements** You can now set up a database mirroring system using SQL statements instead of specifying mirroring settings on the database server command line. You must upgrade or rebuild existing databases to use this feature.

See:

- "CREATE MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
- "ALTER MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
- "SET MIRROR OPTION statement" [*SQL Anywhere Server - SQL Reference*]
- "DROP MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
- MIRROR SERVER clause added to the "COMMENT statement" [*SQL Anywhere Server - SQL Reference*]
- "SYSMIRROROPTION system view" [*SQL Anywhere Server - SQL Reference*]
- "SYSMIRRORSERVER system view" [*SQL Anywhere Server - SQL Reference*]
- "SYSMIRRORSERVEROPTION system view" [*SQL Anywhere Server - SQL Reference*]
- "MirrorRole database property" [*SQL Anywhere Server - Database Administration*]
- Extended database properties: MirrorServerState, MirrorState
- "sa_mirror_server_status system procedure" [*SQL Anywhere Server - SQL Reference*]
- "Database mirroring behavior changes and deprecated features" on page 31

- **Host connection parameter** The new Host connection parameter takes a host name (or IP address) and optional port number that tells the client where to find the database server. This connection parameter is now the recommended way to connect to database servers running on a different computer than the client. See "Host connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Enhancements to automatic statistics management** SQL Anywhere 12 includes a statistics governor that improves the automatic maintenance of statistics on database columns. The health and usefulness of each statistic in the database is automatically evaluated, and required maintenance is performed so that the statistics are self-monitored and self-healing. Statistics maintenance is performed in the background and does not create a significant load on database server performance. See "How the statistics governor maintains statistics" [*SQL Anywhere Server - SQL Usage*].

  The sa_server_option system procedure now supports the following options to help you manage statistics: DropBadStatistics, DropUnusedStatistics, and StatisticsCleaner. See "sa_server_option system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Sequences** SQL Anywhere now supports the generation of sequences. Sequences can be used by applications to generate unique key values. Using sequence values can help applications prevent concurrency and performance issues.

  You can also create, edit, and manage sequences using the SQL Anywhere plug-in in Sybase Central. For example, use the **Create Sequence Generator Wizard** to create a new sequence in the database.

See also:

- "Using a sequence to generate unique values" [*SQL Anywhere Server - SQL Usage*]
- "CREATE SEQUENCE statement" [*SQL Anywhere Server - SQL Reference*]
- "ALTER SEQUENCE statement" [*SQL Anywhere Server - SQL Reference*]
- "DROP SEQUENCE statement" [*SQL Anywhere Server - SQL Reference*]
- "SYSSEQUENCE system view" [*SQL Anywhere Server - SQL Reference*]
- "SYSSEQUENCEPERM system view" [*SQL Anywhere Server - SQL Reference*]
- SEQUENCE clause, "COMMENT statement" [*SQL Anywhere Server - SQL Reference*]
- GRANT USAGE ON SEQUENCE syntax, "GRANT statement" [*SQL Anywhere Server - SQL Reference*]
- REVOKE USAGE ON SEQUENCE syntax, "REVOKE statement" [*SQL Anywhere Server - SQL Reference*]

You must upgrade or rebuild existing databases to use sequences.

- **Multiprogramming level enhancements** The network database server (dbsrv12) now automatically controls its multiprogramming level by default. This behavior allows the database server to improve its throughput and adapt to workload changes without DBA intervention.

  When the database server starts, it creates a pool of workers that are used to service requests. The number of workers is the current multiprogramming level of the server. The pool has minimum and maximum limits, and the current multiprogramming level is always within those limits. The DBA can change the minimum and maximum values at start up by using database server options or while the database server is running by using the sa_server_option system procedure.

  The following options have been added to allow you to control the database server's multiprogramming level:

Database server option	sa_server_option value	Description
"-gn dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]	CurrentMultiProgrammingLevel	Sets the initial multiprogramming level of the database server.
"-gna dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]	AutoMultiProgrammingLevel	Turns on and off dynamic tuning of the database server's multiprogramming level.
"-gnh dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]	MaxMultiprogrammingLevel	Sets the maximum number of tasks that the database server can execute concurrently.

Database server option	sa_server_option value	Description
"-gnl dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]	MinMultiProgrammingLevel	Sets the minimum number of tasks that the database server can execute concurrently.
"-gns dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]	AutoMultiProgrammingLevelStatistics	Controls whether statistics about the automatic changes to the multiprogramming level appear in the database server message log.

For more information about the multiprogramming level in SQL Anywhere, see "Configuring the database server's multiprogramming level" [*SQL Anywhere Server - Database Administration*].

- **Immediate materialized views now support outer joins** Materialized views containing OUTER JOINs in their definitions can now be declared immediate. See "Restrictions on materialized views" [*SQL Anywhere Server - SQL Usage*].

- **Selecting from DML statements** You can now specify a DML statement in the FROM clause of the SELECT statement. This feature allows you to write SQL queries over a derived table populated by the rows modified by an UPDATE, INSERT, DELETE, or MERGE statement and return values from these updated rows to the application.

  The most common use of this feature is to verify or validate the values of rows that have been modified by the application. Previously, the only way to accomplish this would be through the use of a trigger and multiple SQL statements. See "FROM clause" [*SQL Anywhere Server - SQL Reference*] and "Executing a SELECT over a DML statement" [*SQL Anywhere Server - SQL Usage*].

- **Full text search feature now supports external prefilter and term breaker libraries** A new API has been added to allow you to connect to external external prefilter and term breaker libraries when creating and updating full text indexes. This means you can can take document formats like XML, PDF, and Word and remove unwanted terms and content before indexing their content. Some sample prefilter and term breaker libraries are included to help you design your own, or you can use 3rd party libraries. See "External term breaker and prefilter libraries" [*SQL Anywhere Server - SQL Usage*].

  If Microsoft Office is installed on the system running the database server then IFilters for Office documents such as Word and Excel are available. If the server has Acrobat Reader installed, then a PDF IFilter is likely available.

  The PREFILTER EXTERNAL NAME clause and TERM BREAKER EXTERNAL NAME clause have been added to the ALTER TEXT CONFIGURATION statement to allow you to specify the name and location of your external libraries. See "ALTER TEXT CONFIGURATION statement" [*SQL Anywhere Server - SQL Reference*].

Additionally, a new API is provided for working with external prefilter and term breaker libraries.

The ISYSTEXTCONFIG system table has been modified to hold information about the entry points and the external libraries used for tokenizing and/or prefiltering. Specifically, the existing prefilter column data type has changed to be a LONG VARCHAR to hold the entry points and library name for an external prefilter library. A new LONG VARCHAR column, external_term_breaker, has been added to hold the entry points and library name for an external term breaker library. See "SYSTEXTCONFIG system view" [*SQL Anywhere Server - SQL Reference*].

You must upgrade your database to use external prefilter and term breaker libraries.

- **Checksum enhancements** The database server now determines whether to create write checksums for databases pages (checksums that are created only when the pages are written to disk) based on the database version. By default, version 10 and 11 databases do not have global checksums enabled, and version 12 databases have global checksums enabled. When you start an older database on a version 12 database server, the default behavior of the database server is to enable write checksums. For version 12 databases, the database server's default behavior is to not enable write checksums because by default version 12 databases have global checksums enabled. See "Checksums enabled by default for new databases" on page 29.

  You can use the CHECKSUM clause of the START DATABASE statement or the -wc option when starting a database or database server to change the database server behavior for write checksums. See "-wc dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*], "-wc dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*], and "START DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

  You can disable checksums for a database using the CHECKSUM clause of the ALTER DATABASE statement. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

Database connections

Following is a list of enhancements made to database connections in SQL Anywhere version 12.0.0.

- **Temporary connections are named** Temporary connections are used to perform operations such as running backups or initializing databases. You can get information about temporary connections by using the sa_conn_info system procedure, the sa_conn_list system procedure and the Name and ParentConnection connection properties. See:

  - "Temporary connections" [*SQL Anywhere Server - Database Administration*]
  - "Name connection property" [*SQL Anywhere Server - Database Administration*]
  - "ParentConnection connection property" [*SQL Anywhere Server - Database Administration*]
  - "sa_conn_info system procedure" [*SQL Anywhere Server - SQL Reference*]
  - "sa_conn_list system procedure" [*SQL Anywhere Server - SQL Reference*]

- **Connection pooling** The ConnectionPool connection parameter controls the behavior of client connection pooling. See "ConnectionPool (CPOOL) connection parameter" [*SQL Anywhere Server - Database Administration*] and "SQL Anywhere connection pooling" [*SQL Anywhere Server - Database Administration*].

- **Escape connection parameter supported for ODBC data sources** By default the ODBC driver uses the tilde (~) as an escape character, but some applications assume that the escape character is the backslash (\). You can use the Escape connection parameter to specify the escape character for your application. See "Escape connection parameter (ODBC)" [*SQL Anywhere Server - Database Administration*].

  In addition, you can specify the Escape connection parameter in the **Connect** window.

Backup and recovery

Following is a list of backup and recovery enhancements introduced in SQL Anywhere version 12.0.0.

- **Restoring archive backups created with version 12** Archive backups created with version 12 database servers cannot be restored with version 11 or earlier database servers.
- **Free pages are eliminated in archive backups** By default, archive backups skip some free pages, which can result in smaller and potentially faster backups. Free page elimination has no affect on the back up of transaction log files because transaction log files do not contain free pages. So, databases with large transaction log files may not benefit as much from free page elimination as databases with small transaction log files. You can control this behavior using the BACKUP statement's FREE PAGE ELIMINATION clause or using the **Backup Database Wizard**. See "BACKUP statement" [*SQL Anywhere Server - SQL Reference*].

  When you are restoring a strongly-encrypted database that was backed up with free page elimination on, the encryption key for the database must be specified. You can specify the encryption key using the RESTORE DATABASE statement's KEY clause or using the **Restore Database Wizard**.

Security

Following is a list of security enhancements introduced in SQL Anywhere version 12.0.0.

- **FIPS now available on 64-bit Windows and 32- and 64-bit Linux operating systems** FIPS-certified security algorithms can now be used on 64-bit Windows and 32 and 64-bit Linux operating systems.

  For a list of platforms on which the FIPS algorithms are supported, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

  For information about using FIPS, see "FIPS-approved encryption technology" [*SQL Anywhere Server - Database Administration*].

Database permissions and authorities

The following list contains the new or enhanced permissions and authorities in SQL Anywhere. You must upgrade or rebuild your database to make use of these changes. See "Upgrading SQL Anywhere" on page 304.

- **SYS_SPATIAL_ADMIN_ROLE group** Membership in this group allows users to create, alter, or drop spatial reference systems and units of measure. See "Spatial permissions" [*SQL Anywhere Server - Spatial Data Support*].

Database utilities

Following is a list of enhancements made to database utilities in SQL Anywhere version 12.0.0.

- **Database page sizes can now be specified in kilobytes** The Extraction utility (dbxtract), Initialization utility (dbinit), and Unload utility (dbunload) now allow you to specify the database page size in units of bytes or kilobytes. Previously, you could only specify the page size in bytes. See:
  - "Extraction utility (dbxtract)" [*SQL Remote*]
  - "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*]
  - "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*]

- **File Hiding utility (dbfhide) enhancements** On Windows, the File Hiding utility (dbfhide) now supports using a Windows encryption API to obfuscate configuration files. The utility also supports the -q option so it can be run quietly. See "File Hiding utility (dbfhide)" [*SQL Anywhere Server - Database Administration*].

- **Server Licensing utility (dblic) enhancements** In previous releases if you wanted to change your edition of SQL Anywhere, you had to uninstall the software and then reinstall with the new license key. You can now use the -k option for the Server Licensing utility (dblic) to change your edition of SQL Anywhere when you receive a new license key from Sybase iAnywhere. See "Server Licensing utility (dblic)" [*SQL Anywhere Server - Database Administration*].

- **Service utility (dbsvc) enhancements** When specifying dependencies for services with the -rs option of the dbsvc utility on Windows, you can now specify the display name or the actual service name. See "Service utility (dbsvc) for Windows" [*SQL Anywhere Server - Database Administration*].

- **Support utility (dbsupport) enhancements** The -ac option allows you to add a comment that is included in the error report. The -af option allows you to include a file with the error report submission. See "Support utility (dbsupport)" [*SQL Anywhere Server - Database Administration*].

- **Unload utility (dbunload) enhancements** The dbunload utility now supports the following options:
  - The -kd option unloads a database into a single dbspace. See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].
  - The -qr option prevents progress messages from being created and displayed when loading tables and creating indexes. See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

Database options

Following is a list of enhancements made to database options in SQL Anywhere version 12.0.0.

- **uuid_has_hypens option** The uuid_has_hyphens controls the formatting of unique identifier values when they are converted to strings. This feature was removed in version 11, but is now reinstated. See "uuid_has_hyphens option" [*SQL Anywhere Server - Database Administration*].

- **blocking_others_timeout option** This option specifies the amount of time that another connection can block on the current connection's row and table locks before the current connection is rolled back. This option can be used to prevent a low priority task from blocking other connections for longer than the specified time. See "blocking_others_timeout option" [*SQL Anywhere Server - Database Administration*].

- **http_connection_pool_basesize option** This option specifies a baseline size for connection pools used by HTTP. See "http_connection_pool_basesize option" [*SQL Anywhere Server - Database Administration*].

- **http_connection_pool_timeout option** This option specifies the maximum duration that an unused connection may be retained within an HTTP connection pool. See "http_connection_pool_timeout option" [*SQL Anywhere Server - Database Administration*].

- **progress_messages option** This option controls whether progress messages are sent from the database server to the client. You can set this option as a temporary option for the utility database. See "progress_messages option" [*SQL Anywhere Server - Database Administration*], and "Allowed statements for the utility database" [*SQL Anywhere Server - Database Administration*].

  By default Interactive SQL shows the progress messages in the **Messages** pane. You can also set the SQL Anywhere database option progress_messages by choosing **Tools** » **Options** » **SQL Anywhere** » **Commands** and selecting **Show Progress Messages**. When selected, this option sets the database progress_messages option to Formatted. When cleared the database progress_messages option is set to Off. By default, the **Show Progress Messages** option is selected.

- **timestamp_with_time_zone_format option** This option controls how TIMESTAMP WITH TIME ZONE values are converted to strings. See "timestamp_with_time_zone_format option" [*SQL Anywhere Server - Database Administration*].

- **reserved_keywords option** This option turns on individual keywords. See "reserved_keywords option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_asbinary_format option** Controls how spatial values are converted from a geometry to binary. See "st_geometry_asbinary_format option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_astext_format option** Controls how spatial values are converted from a geometry to text. See "st_geometry_astext_format option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_asxml_format option** Controls how spatial values are converted from a geometry to XML. See "st_geometry_asxml_format option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_describe_type option** Controls how spatial values are described. See "st_geometry_describe_type option" [*SQL Anywhere Server - Database Administration*].

- **st_geometry_on_invalid option** Controls the behavior when a geometry fails basic validation. See "st_geometry_on_invalid option" [*SQL Anywhere Server - Database Administration*].

Database server options

Following is a list of enhancements made to database server options in SQL Anywhere version 12.0.0.

- **-xm option** The -xm database server option controls how often the database server checks for new IP addresses. If a computer is connected to a new network or it is disconnected from an existing network and the -xm option is specified, the database server starts listening on the new network when the change is detected. See "-xm dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Properties and Performance Monitor statistics

Following is a list of enhancements made to properties and Performance Monitor statistics in SQL Anywhere version 12.0.0.

- **Connection properties** The following connection properties have been added in this release:

  - blocking_others_timeout
  - http_connection_pool_basesize
  - http_connection_pool_timeout
  - ParentConnection
  - Progress
  - progress_messages
  - QueryRowsFetched
  - reserved_keywords
  - st_geometry_asbinary_format
  - st_geometry_astext_format
  - st_geometry_asxml_format
  - st_geometry_describe_type
  - st_geometry_on_invalid
  - TempFilePages
  - timestamp_with_time_zone_format

  For more information about these properties, see "Connection properties" [*SQL Anywhere Server - Database Administration*].

- **Database properties** The following database properties have been added in this release:

  - ConnPoolCachedCount
  - ConnPoolHits
  - ConnPoolMisses
  - DriveBus
  - DriveModel
  - HasTornWriteFix
  - HttpConnPoolCachedCount
  - HttpConnPoolHits
  - HttpConnPoolMisses
  - HttpConnPoolSteals
  - LastCheckpointTime
  - MirrorRole
  - QueryRowsFetched
  - SynchronizationSchemaChangeActive
  - WriteChecksums

  For more information about these properties, see "Database properties" [*SQL Anywhere Server - Database Administration*].

- **Database server properties** The following database server properties have been added in this release:

  - AutoMultiProgrammingLevel
  - AutoMultiProgrammingLevelStatistics
  - CurrentMultiProgrammingLevel
  - IPAddressMonitorPeriod
  - IsPortableDevice
  - MaxMultiProgrammingLevel
  - MinMultiProgrammingLevel
  - ObjectType
  - ThreadDeadlocksAvoided
  - ThreadDeadlocksReported

  For more information about these properties, see "Database server properties" [*SQL Anywhere Server - Database Administration*].

System procedures and functions

Following is a list of system procedure and function enhancements added in SQL Anywhere version 12.0.0.

- **New sa_text_index_vocab_nchar system procedure** This new system procedure is for use with NCHAR text indexes and is equivalent to sa_text_index_vocab. See "sa_text_index_vocab_nchar system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New sa_copy_cursor_to_temp_table system procedure** Copies the contents of a cursor to a temporary table. See "sa_copy_cursor_to_temp_table system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New sa_describe_cursor system procedure** Returns the name and data types of columns in a cursor. While this information can be retrieved from various client programming interfaces, it was not previously accessible within stored procedures. See "sa_describe_cursor system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New sa_install_feature system procedure** Installs additional features that were not present in the database when SQL Anywhere was installed. See "sa_install_feature system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New sa_list_cursors system procedure** Returns a result set with one row for each of the cursors maintained by the database server for this connection. The result set gives the cursor name, a value indicating whether the cursor is currently open, and other meta information. See "sa_list_cursors system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New sa_mirror_server_status system procedure** This procedure reports the status of copy nodes in the tree below the database server on which the procedure is run. See "sa_mirror_server_status system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New sa_reserved_words system procedure** This procedure returns a list of SQL Anywhere reserved words. See "sa_reserved_words system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the sa_server_option system procedure** The following options have been added to the sa_server_option system procedure:

  - AutoMultiProgrammingLevel
  - AutoMultiProgrammingLevelStatistics
  - CurrentMultiProgrammingLevel
  - DropBadStatistics
  - DropUnusedStatistics
  - IPAddressMonitorPeriod
  - MaxMultiProgrammingLevel
  - MinMultiProgrammingLevel
  - StatisticsCleaner

  See "sa_server_option system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the sa_table_page_usage system procedure** When the progress_messages database option is set to Raw or Formatted, this system procedure periodically sends progress messages while it is running. See "sa_table_page_usage system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the xp_read_file system procedure** The xp_read_file system procedure now includes an optional parameter that allows you to specify lazy reads. When you specify this optional parameter and its value is not zero, the file is read and then immediately unlocked. See "xp_read_file system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the xp_startsmtp system procedure** The xp_startsmtp system procedure supports four new parameters: trusted_certificates, certificate_company, certificate_unit, and certificate_name. These parameters allow you to send mail using secure SMTP. You must upgrade your database to use this feature. See "xp_startsmtp system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the openxml system procedure** The openxml system procedure now supports USING FILE and USING VALUE clauses that allow you to load data from a file or expression of CHAR, NCHAR, BINARY, or LONG BINARY type, or BLOB string, respectively. See "openxml system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New MEDIAN function** Computes the median of a numeric expression for a set of rows. See "MEDIAN function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to the HASH function** The HASH function now accepts the CRC32 algorithm type. See "HASH function [String]" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the BIT_OR, BIT_AND, and BIT_XOR functions** The BIT_OR, BIT_AND, and BIT_XOR functions now support integer values and binary values. Also, BIT_OR, BIT_AND, and BIT_XOR functions can now be used in parallel execution plans.

  See:

  - "BIT_OR function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]
  - "BIT_AND function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]
  - "BIT_XOR function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]

- **Enhancements to the DATEADD, DATEDIFF, DATEPART, and DATENAME functions** The microsecond date part and the TIMESTAMP WITH TIME ZONE data type are now supported by the DATEADD, DATEDIFF, DATEPART, and DATENAME functions. See:

  - "DATEADD function [Date and time]" [*SQL Anywhere Server - SQL Reference*]
  - "DATEDIFF function [Date and time]" [*SQL Anywhere Server - SQL Reference*]
  - "DATEPART function [Date and time]" [*SQL Anywhere Server - SQL Reference*]
  - "DATENAME function [Date and time]" [*SQL Anywhere Server - SQL Reference*]

- **Enhancements to the DB_EXTENDED_PROPERTY function** You can now use the DB_EXTENDED_PROPERTY function with the MirrorServerState and MirrorState properties to determine the synchronization and connection status of a mirror server. See "DB_EXTENDED_PROPERTY function [System]" [*SQL Anywhere Server - SQL Reference*].

- **New HTTP_RESPONSE_HEADER function** Returns the value of an HTTP response header. See "HTTP_RESPONSE_HEADER function [HTTP]" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to HTTP_VARIABLE function** You can now use the @BINARY attribute to return an x-www-form-urlencoded binary data value. See "HTTP_VARIABLE function [HTTP]" [*SQL Anywhere Server - SQL Reference*].

- **New ISENCRYPTED function** Determines if a string is encrypted. See "ISENCRYPTED function [System]" [*SQL Anywhere Server - SQL Reference*].

- **New NEXT_HTTP_RESPONSE_HEADER function** Gets the next HTTP response header name. See "NEXT_HTTP_RESPONSE_HEADER function [HTTP]" [*SQL Anywhere Server - SQL Reference*].

- **New SWITCHOFFSET function** Returns a TIMESTAMP WITH TIME ZONE value that is converted from its original time zone offset to the specified time zone offset. See "SWITCHOFFSET function [Date and time]" [*SQL Anywhere Server - SQL Reference*].

- **New SYSDATETIMEOFFSET function** Returns the current date, time, and time zone offset of the database server. See "SYSDATETIMEOFFSET function [Date and time]" [*SQL Anywhere Server - SQL Reference*].

- **New TODATETIMEOFFSET function** Converts a TIMESTAMP value to a TIME STAMP WITH TIME ZONE value using the specified time zone offset. See "TODATETIMEOFFSET function [Date and time]" [*SQL Anywhere Server - SQL Reference*].

- **New COUNT_BIG function** Counts the number of rows in a group depending on the specified parameters. See "COUNT_BIG function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].

SQL statements

Following is a list of SQL enhancements introduced in SQL Anywhere version 12.0.0.

- **Improved qualification for named index hints** A PRIMARY KEY index and a FOREIGN KEY index on a table can have the same name. When this is true, a named index hint can not be unambiguously specified. The specification of a named index hint has been extended to allow the qualification of a hinted index as a PRIMARY KEY index or a FOREIGN KEY index. See "INDEX clause, FROM clause" [*SQL Anywhere Server - SQL Reference*].

- **New IS DISTINCT FROM and IS NOT DISTINCT FROM search conditions** The IS DISTINCT FROM and IS NOT DISTINCT FROM search conditions allow you to control the evaluation of equality in the case of NULLs. See "IS DISTINCT FROM and IS NOT DISTINCT FROM search conditions" [*SQL Anywhere Server - SQL Reference*].

- **CREATE SYNCHRONIZATION PROFILE statement** This statement creates a SQL Anywhere synchronization profile. A synchronization profile defines how a SQL Anywhere database synchronizes with the MobiLink server. See "CREATE SYNCHRONIZATION PROFILE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

- **New WITH NULLS NOT DISTINCT clause, CREATE INDEX statement** A new clause, WITH NULLS NOT DISTINCT, has been added to the CREATE INDEX statement for use when creating UNIQUE indexes. This clause allows you to specify that NULLs in index keys are not unique. You must upgrade or rebuild existing databases to use this feature. See the UNIQUE clause of the "CREATE INDEX statement" [*SQL Anywhere Server - SQL Reference*].

- **Back quote identifier delimiter** Back quotes can now be used as an identifier delimiter. See "Identifiers" [*SQL Anywhere Server - SQL Reference*].

- **New SAVE OPTION VALUES clause for ALTER TEXT CONFIGURATION statement** Use this new clause to change the values of the date_format, time_format, timestamp_format, and timestamp_with_time_zone_format options saved with a text configuration object. See "ALTER TEXT CONFIGURATION statement" [*SQL Anywhere Server - SQL Reference*].

- **ALTER SERVER and CREATE SERVER statements** The new IQODBC and IQJDBC server classes allow you to specify a Sybase IQ server as a remote connection. See "ALTER SERVER statement" [*SQL Anywhere Server - SQL Reference*] and "CREATE SERVER statement" [*SQL Anywhere Server - SQL Reference*].

- **New LOCK FEATURE statement** This statement prevents other concurrent connections from using a database server feature. See "LOCK FEATURE statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the BEGIN, CREATE VARIABLE, and DECLARE statements** Variable declarations can now include an initial starting value for the variable. See:

  - "BEGIN statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE VARIABLE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DECLARE statement" [*SQL Anywhere Server - SQL Reference*]

- **Enhancements to the LOAD TABLE statement** The LOAD TABLE statement now supports the load-option clause, which allows you to control how data is loaded. It also supports the value XML in the FORMAT clause. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **New IF NOT EXISTS clause** With the new IF NOT EXISTS clause, no changes are made and an error is not returned if the named object already exists. See:

  - "CREATE INDEX statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PUBLICATION statement [MobiLink] [SQL Remote]" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE SPATIAL REFERENCE SYSTEM statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE TEXT INDEX statement" [*SQL Anywhere Server - SQL Reference*]

- **New IF EXISTS clause** The new IF EXISTS clause allows you to specify that you do not want an error returned when the DROP statement attempts to remove a database object that does not exist. See:
  - "DROP EVENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP FUNCTION statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP INDEX statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP PUBLICATION statement [MobiLink] [SQL Remote]" [*SQL Anywhere Server - SQL Reference*]
  - "DROP SPATIAL REFERENCE SYSTEM statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP SPATIAL UNIT OF MEASURE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP SYNCHRONIZATION PROFILE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]
  - "DROP TABLE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP TRIGGER statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP VARIABLE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP VIEW statement" [*SQL Anywhere Server - SQL Reference*]
- **CASE statement enhancements** CASE statements are now supported in Transact-SQL procedures and batches.
- **New OR REPLACE clause** The new OR REPLACE clause allows you to create or replace a profile or variable of the same name. See:
  - "CREATE FUNCTION statement (external procedures)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE FUNCTION statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PROCEDURE statement (external procedures)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PROCEDURE statement [T-SQL]" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE SEQUENCE statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE SPATIAL REFERENCE SYSTEM statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE SYNCHRONIZATION PROFILE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE TRIGGER statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE VARIABLE statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE VIEW statement" [*SQL Anywhere Server - SQL Reference*]
- **New LIMIT clause support for SELECT statements** You can now specify row counts and offsets in a SELECT statement using the new LIMIT clause. See "SELECT statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to the SET OPTION statement** Syntax 1 of the SET OPTION statement now supports setting an option using the contents of a variable. See "SET OPTION statement" [*SQL Anywhere Server - SQL Reference*].

- **New IF [NOT] OF search condition** The IF [NOT] OF *type-expression* search condition has been added. See "Search conditions" [*SQL Anywhere Server - SQL Reference*] and "Spatial data type syntax based on ANSI SQL UDTs" [*SQL Anywhere Server - Spatial Data Support*].

- **INSERT statement enhancements** The following enhancements have been made to the INSERT statement. See "INSERT statement" [*SQL Anywhere Server - SQL Reference*].

  - **Support for more than one list of values** An INSERT statement can now contain more than one list of values, allowing several rows to be inserted at once. For example:

    ```
    INSERT INTO T (c1,c2,c3)
    VALUES (1,10,100), (2,20,200), (3,30,300);
    ```

  - **Support for inserting rows with all default values** SQL Anywhere allows the VALUES clause to contain specified values for a subset of the columns in the table. All unspecified columns are given default values as specified for each column by means of DEFAULT, NULL and COMPUTE clauses of the CREATE TABLE statements. Previously, the database server required that you specify input values for at least one of the columns in the table.

    Now, however, all columns can be given their default values by using either of the following syntax extensions:

    INSERT [**INTO**] *table-name options* **DEFAULT VALUES** ...

    INSERT [**INTO**] *table-name* **( )** *options* **VALUES ( )** [**, ( )** ...]

    Specifying DEFAULT VALUES or VALUES is semantically equivalent to using the following syntax, where the number of default entries is equal to the number of columns in the table:

    INSERT [**INTO**] *table-name* **VALUES(default, default, ..., default)**

    The DEFAULT VALUES clause is part of the SQL/2008 standard, whereas the VALUES clause is a vendor extension.

- **New START SERVER statement** The START SERVER statement has been added. It should be used instead of the START ENGINE statement, which is now deprecated. See "START SERVER statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

- **New STOP SERVER statement** The STOP SERVER statement has been added. It should be used instead of the STOP ENGINE statement, which is now deprecated. See "STOP SERVER statement" [*SQL Anywhere Server - SQL Reference*].

- **New CHECKSUM clause for the ALTER DATABASE statement** The CHECKSUM OFF clause lets you disable global checksums for databases. Global checksums are enabled by default for new version 12 databases. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **DELETE statement enhancement** The DELETE statement now supports correlation names. See "DELETE statement" [*SQL Anywhere Server - SQL Reference*].

Data types

Following is a list of enhancements to data types introduced in SQL Anywhere version 12.0.0.

- **TIMESTAMP WITH TIME ZONE data type** Stores a point in time with a time zone offset. See "TIMESTAMP WITH TIME ZONE data type" [*SQL Anywhere Server - SQL Reference*].

  The alias DATETIMEOFFSET is also supported. See "DATETIMEOFFSET data type" [*SQL Anywhere Server - SQL Reference*].

- **CHAR, NCHAR, NVARCHAR, and VARCHAR data types support 32767 character length** The NCHAR and NVARCHAR data types can now be declared up to 32767 characters. The CHAR and VARCHAR data types with character-length semantics can now be declared up to 32767 characters. If the described byte length in the client character set for the NCHAR, NVARCHAR, CHAR, or VARCHAR data types with character length semantics would be more than 32767, then these types are described as the LONG NVARCHAR or LONG VARCHAR data type. The described byte length for CHAR and VARCHAR data types with byte length semantics now accounts for the maximum possible character set expansion when converting values to the client character set. If the resulting length including the maximum possible expansion is greater than 32767, the type is described as LONG VARCHAR. There is no expansion if the client is using a single byte character set, or the client character set and the database character set encoding are the same. See:

  - "CHAR data type" [*SQL Anywhere Server - SQL Reference*]
  - "NCHAR data type" [*SQL Anywhere Server - SQL Reference*]
  - "NVARCHAR data type" [*SQL Anywhere Server - SQL Reference*]
  - "VARCHAR data type" [*SQL Anywhere Server - SQL Reference*]

- **Data types for spatial data** Several new data types have been added to support spatial data. See "Accessing and manipulating spatial data" [*SQL Anywhere Server - Spatial Data Support*].

Programming interfaces

Following is a list of enhancements to programming interfaces introduced in SQL Anywhere version 12.0.0.

- **Web services performance enhancements** HTTP services now support automatic connection pooling to maximize the effect of plan caching and benefit from the potential performance improvement. See "Developing web service applications in an HTTP web server" [*SQL Anywhere Server - Programming*].

- **Support for customized ODBC driver names** To facilitate the installation and registration of multiple independent copies of the SQL Anywhere ODBC driver on a client system, you can now assign a customized name to the ODBC driver. See "Configuring the ODBC driver" [*SQL Anywhere Server - Programming*].

- **ANSI-only ODBC driver for Unix** Versions of Unix ODBC driver managers that define SQLWCHAR as 32-bit (UTF-32) quantities cannot be used with the SQL Anywhere ODBC driver that supports wide calls since this driver is built for 16-bit SQLWCHAR. For these cases, an ANSI-only version of the SQL Anywhere ODBC driver is provided. This version of the ODBC driver does not support the wide call interface (SQLConnectW for example). See "Using a UTF-32 ODBC driver manager on Unix" [*SQL Anywhere Server - Programming*].

- **DBTools: no_reload_status bit field added to an_unload_db structure** A new bit field, no_reload_status, has been added to the an_unload_db structure. You can use no_reload_status to suppress table and index progress messages on reload. See "an_unload_db structure" [*SQL Anywhere Server - Programming*].

- **New SQL Anywhere TYPE-2 JDBC driver** A new TYPE-2 JDBC driver is now available for JDBC applications to use when connecting to SQL Anywhere. Unlike the iAnywhere JDBC driver, which sits on top of ODBC and can be used to connect to a variety of servers via ODBC, the SQL Anywhere JDBC driver connects to SQL Anywhere only and does not require the SQL Anywhere ODBC driver to be installed or registered.

  The SQL Anywhere JDBC driver comes with a JDBC 3.0 driver and a JDBC 4.0 driver.

  Note
  If you currently use the iAnywhere JDBC driver, it is strongly recommended that you change to the new SQL Anywhere JDBC driver.

  The JDBC 4.0 driver automatically loads and registers itself.

  To use the version 3.0 SQL Anywhere JDBC driver, you must load the sybase.jdbc.sqlanywhere.IDriver class that implements the java.sql.Driver interface and registers the SQL Anywhere JDBC driver with the JDBC driver manager. Once loaded, connections using the SQL Anywhere JDBC driver can be made by using the URL jdbc:sqlanywhere:*connection-string-parameters*. The *connection-string-parameters* are the standard connection parameters required to connect to SQL Anywhere. Note that the application no longer needs to specify DRIVER= or DSN= in *connection-string-parameters* when using the SQL Anywhere JDBC driver. See "JDBC support" [*SQL Anywhere Server - Programming*].

- **New support for JDBC Statement class methods** Previously, the JDBC drivers only supported the addBatch and executeBatch methods of the PreparedStatement class. The JDBC drivers now also support the addBatch, clearBatch, and executeBatch methods of the Statement class. Due to the fact that the JDBC specification is unclear on the behavior of the executeBatch method of the Statement class, the following notes should be considered when using this method with the SQL Anywhere JDBC drivers:

  1. Processing of the batch stops immediately upon encountering a SQL exception or result set. If processing of the batch stops, then a BatchUpdateException will be thrown by the executeBatch method. Calling the getUpdateCounts method on the BatchUpdateException will return an integer array of row counts where the set of counts prior to the batch failure will contain a valid non-negative update count; while all counts at the point of the batch failure and beyond will contain a

-1 value. Casting the BatchUpdateCount to a SQLException provides additional details as to why batch processing was stopped.

2. The batch is only cleared when the clearBatch method is explicitly called. As a result, calling the executeBatch method repeatedly will re-execute the batch over and over again. In addition, calling execute(sql_query) or executeQuery(sql_query) correctly executes the specified SQL query, but does not clear the underlying batch. Hence, calling the executeBatch method followed by execute(sql_query) followed by the executeBatch method again will execute the set of batched statements, then execute the specified SQL query, and then execute the set of batched statements again.

See "JDBC support" [*SQL Anywhere Server - Programming*].

- **RESUME statement returns row counts or the row estimate** The RESUME statement now returns the number of rows in the result set or the row estimate for the next result set in a procedure call. In previous versions, this row count was not available after the RESUME statement. To get the RESUME row count, both the client application and the database server must be SQL Anywhere 12.

  This change affects the following client APIs:

API	Function call or statement affected	Row count returned by
Embedded SQL	RESUME statement	SQLCOUNT field
ODBC	SQLMoreResults function	SQLRowCount function
OLE DB	IMultipleResults::GetResult method	pcRowsAffected output parameter
PHP	sasql_next_result function	sasql_num_rows function
PHP	sasql_stmt_next_result function	sasql_stmt_num_rows function
SQL Anywhere C API	sqlany_get_next_result function	sqlany_num_rows function
SQL Anywhere Ruby API	sqlany_get_next_result function	sqlany_num_rows function
SQL Anywhere Python API	nextset method	rowcount attribute

- **PHP $_SERVER variable now contains all variables required by CGI/1.1** The $_SERVER variable in PHP now contains all variables required by CGI/1.1 when executing an HTTP request. See "The PHP external environment" [*SQL Anywhere Server - Programming*].

  For more information on the variables required by CGI/1.1, see Section 4 of RFC3875: http://www.ietf.org/rfc/rfc3875

- **Fine control over Python data type mapping added to sqlanydb** To control how database types are mapped into Python objects when results are fetched from the database server, conversion callbacks can be registered. See "Database type conversion" [*SQL Anywhere Server - Programming*].

- **New DBLib fill_sqlda_ex function** The fill_sqlda_ex function provides additional functionality over fill_sqlda. In particular, a flag can be provided to preserve the DT_LONGVARCHAR, DT_LONGNVARCHAR and DT_LONGBINARY types when filling the descriptor. See "fill_sqlda_ex function" [*SQL Anywhere Server - Programming*].

- **Silent install feature selection now possible** On Windows, SQL Anywhere components or features may be selected or omitted for install from the command line. For example, specify AT32=1 to select the 32-bit administration tools. See "Using a silent install for deployment" [*SQL Anywhere Server - Programming*].

- **TDS now supports 6 digit precision for time and datetime data** Applications that connect to SQL Anywhere using jConnect 7 and later, or Open Client 15.5 and later, can now retrieve 6 digits of precision when querying time or datetime data.

Catalog changes

Following is a list of catalog changes introduced in SQL Anywhere version 12.0.0.

- **New ISYSSEQUENCE system table and SYSSEQUENCE system view** The ISYSSEQUENCE system table contains one row for each user-defined sequence. See "SYSSEQUENCE system view" [*SQL Anywhere Server - SQL Reference*].

- **New ISYSSEQUENCEPERM system table and SYSSEQUENCEPERM system view** The ISYSSEQUENCEPERM system table records the privileges that users or groups hold on sequences. See "SYSSEQUENCEPERM system view" [*SQL Anywhere Server - SQL Reference*].

- **ISYSTEXTCONFIG system table** The ISYSTEXTCONFIG system table has been modified to hold information about the entry points and the external libraries used for tokenizing and/or prefiltering. Specifically, the existing prefilter column data type has changed to be a LONG VARCHAR to hold the entry points and library name for an external prefilter library. A new LONG VARCHAR column, external_term_breaker, has been added to hold the entry points and library name for an external term breaker library. See "SYSTEXTCONFIG system view" [*SQL Anywhere Server - SQL Reference*].

- **ISYSTABCOL system table: new base_type_str column** This column holds the annotated type string representing the physical type of the column. See "SYSTABCOL system view" [*SQL Anywhere Server - SQL Reference*].

- **ISYSPROCPARM system table: new base_type_str column** This column holds the annotated type string representing the physical type of the parameter. See "SYSPROCPARM system view" [*SQL Anywhere Server - SQL Reference*].

- **ISYSUSERTYPE system table: new base_type_str column** This column holds the annotated type string representing the physical type of the user type. See "SYSUSERTYPE system view" [*SQL Anywhere Server - SQL Reference*].

- **ISYSOBJECT system table: new object_type_str column** This column holds a word description (for example, MAT VIEW) of the value in SYSOBJECT.object_type.

  Previously, the SYSOBJECT view only provided object_type, which gave the object type expressed as a TINYINT. This meant you needed to access either the documentation or the view definition to translate the integer to a word description. Now, you can query the object_type_str column to see the word description of the object. See "SYSOBJECT system view" [*SQL Anywhere Server - SQL Reference*].

Windows Mobile enhancements

Following is a list of Windows Mobile enhancements introduced in SQL Anywhere version 12.0.0.

- **Default stack size for internal execution threads changed for Windows Mobile** The default stack size on Windows Mobile is now 96 KB, the minimum stack size is now 64 KB, and the maximum stack size is now 512 KB. See "-gss dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Unix/Linux enhancements

Following is a list of Unix and Linux enhancements introduced in SQL Anywhere version 12.0.0.

- **Performance Statistics utility (dbstats)** The dbstats utility returns the value of performance statistics on Unix computers. Its behavior is similar to that of the Windows Performance Monitor. See "Performance Statistics utility (dbstats) (Unix)" [*SQL Anywhere Server - Database Administration*].

- **PID file created for Linux services** When a SQL Anywhere service is running on Linux, a PID file is created in the */var/run* directory. See "Service utility (dbsvc) for Linux" [*SQL Anywhere Server - Database Administration*].

Performance enhancements

Following is a list of performance enhancements introduced in version 12.0.0 for which there are no user-visible changes other than performance improvement.

- **Improvements to locking when updating primary rows** Updating a non-key column in a primary row and modifying foreign rows referring to that row no longer interfere with each other. For example, in the sample database, a sales_order row can be added while the corresponding customer address is being updated, without one operation having to wait on the other. See "sa_locks system

procedure" [*SQL Anywhere Server - SQL Reference*] and "Objects that can be locked" [*SQL Anywhere Server - SQL Usage*].

- **Reduced locking at isolation levels 2 and 3** Read locks on individual rows are no longer obtained when a table is locked in share mode (LOCK TABLE...IN SHARE MODE statement). This can result in reduced locking overhead at isolation levels 2 and 3.

- **Improved index performance** SQL Anywhere 12 includes enhanced algorithms and a new on-disk layout to improve the performance of deleting large numbers of clustered sequential values from an index.

- **Improved validation performance** SQL Anywhere 12 includes many enhancements to improve the validation of large databases.

- **Improved request prioritization** SQL Anywhere 12 has been enhanced to boost the priority of I/0 bound requests, which results in better utilization of hardware resources.

- **Improved remote data access performance** SQL Anywhere 12 includes many enhancements to improve remote data access performance, including improved proxy table performance.

- **New cost model** SQL Anywhere 12 includes a CPU cost model that more accurately estimates query execution costs on modern hardware. This behavior may cause changes to access plans for some queries.

- **Enhancement to queries embedded in user-defined functions** SQL queries embedded in user-defined functions can now be in-lined by the query optimizer, which avoids a procedure context switch with each invocation and provides the optimizer with new degrees of freedom with which to optimize the statement.

- **Improvements when converting expressions to different data types** Improvements have been made to the evaluation rules the database server uses when converting an expression to a different data type. The new evaluation rules make the conversion more efficient to execute.

Miscellaneous

Following is a list of miscellaneous enhancements introduced in SQL Anywhere version 12.0.0.

- **New script for making copies of the sample database** The *newdemo.bat* and *newdemo.sh* files are located in the *bin32* or *bin64* directory of your SQL Anywhere installation directory and can be used to create a copy of the sample database that includes all the data from the sample database. This script can be used to re-create the sample database or to make new copies of it with a different name. See "Recreate the sample database (demo.db)" [*SQL Anywhere 12 - Introduction*].

- **New selectivity estimate source type for the query optimizer** A new selectivity estimate source type, JOIN, has been added. This new source type is used by the query optimizer for selectivity estimations of atomic predicates of the form `T.X = R.X`. See "ESTIMATE_SOURCE function [Miscellaneous]" [*SQL Anywhere Server - SQL Reference*].

- **Better handling of overflow errors** Arithmetic operations (+, -, *, /, SUM, AVG) may overflow because the result of the operation can not be represented in the data type. Previously, for expressions of type INT, this overflow returned an error, while for all other data types, the overflow resulted in an undefined value. Now, all arithmetic operations on all types detect overflow and return an error if the result can not be expressed in the data type.

- **Use the ALTER EXTERNAL ENVIRONMENT statement to set the location of dbmlsync** If, during synchronization, the dbmlsync executable cannot be located using the PATH environment variable that the database server is using, you can now use the ALTER EXTERNAL ENVIRONMENT statement to tell the database server the location of the dbmlsync executable. See "ALTER EXTERNAL ENVIRONMENT statement" [*SQL Anywhere Server - SQL Reference*].

- **Back quotes are supported as delimiters** You can now use back quotes (`) to delimit identifiers in SQL Anywhere. See "Identifiers" [*SQL Anywhere Server - SQL Reference*].

- **Japanese Unicode Collation Algorithm (UCA) collation tailoring option** A new Japanese UCA collation tailoring option is now available. You can use it to define a primary-level difference between all Hiragana and Katakana letters. This new tailoring option provides correct equality comparisons of Hiragana and Katakana letters in case-insensitive collations. See "Collation tailoring options" [*SQL Anywhere Server - Database Administration*].

- **Changes to the server messages window and Windows system tray icon** The title bar for the database server messages window now specifies whether you are running a personal server or a network server. The tooltip for the Windows system tray icon also specifies the type of database server. As well, the database server **About** window includes the edition of SQL Anywhere that is running.

- **Information utility (dbinfo) enhancement** The dbinfo utility now returns information about the CHAR collation specification, the CHAR encoding, the NCHAR collation specification, or the NCHAR encoding for a database. See "Information utility (dbinfo)" [*SQL Anywhere Server - Database Administration*].

- **Controlling the amount of address space reserved for non-cache use** The -ch option now leaves more address space for use outside the cache, and the maximum non-AWE cache size on 32-bit operating systems has been reduced. See "-ch dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*] and "-chx dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Enhance priority management for I/O bound compared to CPU bound requests** The database server now dynamically detects requests that are I/O bound and increases their priorities over CPU-bound tasks to increase disk throughput and the use of hardware resources.

- **Improved robustness across power failures on Windows** You can set Windows Registry entries to improve robustness across power failures on systems using certain Intel storage drivers when deploying SQL Anywhere. Failure to set this parameter can result in lost data and corrupted databases in the event of a power failure. To determine if these entries are required as part of your deployment, see "Windows Registry entries" [*SQL Anywhere Server - Programming*].

- **SQL Flagger enhancement** The SQL Flagger now supports the SQL/2008 standard. See "Testing SQL compliance using the SQL Flagger" [*SQL Anywhere Server - SQL Usage*].

- **Progress messages** Some SQL statements now support sending progress messages are sent from the database server to the client. See .

SQL Anywhere behavior changes

Following is a list of behavior changes to SQL Anywhere introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **Checksums enabled by default for new databases** When you create a new database, global checksums are now enabled by default. Global checksums are calculated and verified each time a database page is read or written, and are used to determine whether a database page has been modified on disk. You can control whether global checksums are enabled for a database by using the -s[+ | -] option for the Initialization utility, or by using the CHECKSUM clause of the CREATE DATABASE statement. See:

  - "Checksum enhancements" on page 10
  - "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*]
  - "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*]
  - "START DATABASE statement" [*SQL Anywhere Server - SQL Reference*]

- **Checkpoint log changes** In previous releases, when you stopped a database, SQL Anywhere completely truncated the checkpoint log. In version 12, a history of the checkpoint log usage is maintained in the database and is used to determine an appropriate size for the checkpoint log for the next session.

  Maintaining the checkpoint log across sessions avoids the overhead of allocating it the next time the database is started and avoids file fragmentation that can occur when files are extended. Database files will now be larger when the database is shut down than they were in previous releases, but the additional space is reused for the checkpoint log the next time the database is restarted. See "Understanding the checkpoint log" [*SQL Anywhere Server - Database Administration*].

- **Network database server now allocates the maximum number of workers** In previous releases, at startup the network database server allocated the number of workers that corresponded to the database server multiprogramming level. In version 12 network servers, the number of workers allocated corresponds to the maximum server multiprogramming level. This increase in the number of workers increases the address space requirements of the network database server for worker stacks, and may impact the amount of database cache that the database server can allocate.

  For example, on 32-bit Windows platforms, by default each worker requires 1 MB of address space for its stack. A version 11 network server starting on Windows would require 20 MB of address space for worker stacks, as its default multiprogramming level is 20. However, a version 12 network server starting on Windows requires 80 MB of address space, as the default maximum number of workers is 80. This change does not affect personal servers or Windows Mobile. See "Threading in SQL Anywhere" [*SQL Anywhere Server - Database Administration*].

- **Old statistics are not loaded when a database is rebuilt** When you rebuild a version 11 or earlier database, the LOAD STATISTICS statement silently skips loading old string statistics into the

new database, but upgrades the version of the string statistics. See "LOAD STATISTICS statement" [*SQL Anywhere Server - SQL Reference*].

Upgrading a database (using the Upgrade utility) does not upgrade the version of the string statistics.

- **Positioned DELETE and UPDATE statements** Previously, you could specify a TOP or FIRST clause in a positioned UPDATE or DELETE statement; however, the clauses would be ignored. Now, specifying TOP or FIRST in a positioned UPDATE or DELETE statement returns a syntax error. See "UPDATE (positioned) statement [ESQL] [SP]" [*SQL Anywhere Server - SQL Reference*], and "DELETE (positioned) statement [ESQL] [SP]" [*SQL Anywhere Server - SQL Reference*].

- **JDBC drivers now report CHAR and VARCHAR instead of just CHAR for both** Previously, when an application connected using the iAnywhere JDBC driver and attempted to describe the metadata of a table or result set that contains a CHAR column, the metadata would return the type name of the column as CHAR but the SQL type would still come back as Types.VARCHAR. If a JDBC application wanted to get the JDBC driver to return the SQL type of CHAR columns as Types.CHAR, then the application was required to set the odbc_distinguish_char_and_varchar database option. Now, the new SQL Anywhere JDBC driver and the deprecated iAnywhere JDBC driver return the type name CHAR and SQL type Types.CHAR for table and result set columns of type CHAR, and the type name VARCHAR and the SQL type Types.VARCHAR for columns of type VARCHAR regardless of the database option setting.

- **Changes to CURRENT UTC TIMESTAMP and UTC TIMESTAMP special values** The underlying data type for the CURRENT UTC TIMESTAMP special value, and the default value UTC TIMESTAMP special value, is now TIMESTAMP WITH TIME ZONE. If these values are used with columns defined as TIMESTAMP, the time zone offset will be truncated and no difference in behavior should be noticeable. However, if these values are used with CHAR or VARCHAR columns, the offset will result in different values being generated than before. See "CURRENT UTC TIMESTAMP special value" [*SQL Anywhere Server - SQL Reference*] and "UTC TIMESTAMP special value" [*SQL Anywhere Server - SQL Reference*].

- **Personal server no longer starts TCP/IP by default** The personal database server only starts the shared memory protocol by default. If you want to use the TCP/IP protocol, you must specify it using the -x server option when starting the personal database server. See "-x dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*] and "Selecting communications protocols" [*SQL Anywhere Server - Database Administration*].

- **TCP/IP connections** If you are connecting over TCP/IP, the database server name (specified by the ServerName (SERVER) connection parameter) is no longer mandatory if a host name is provided with the HOST connection parameter. See "Host connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Host (IP) protocol option** In previous releases, the Host protocol option indicated one or more hosts on which the database server may be running, but was considered a hint to the client library. If the database server was not found on those hosts, a network broadcast was done to find the database server. In this release, if the Host option is specified, only the specified hosts are searched for the database server and the client does not broadcast to find the database server by default.

  This behavior is equivalent to setting the DoBroadcast protocol option to Direct. If the database server is running on a computer other than the ones specified by the HOST protocol option, it is not found. If

you want the same behavior as previous releases, specify DoBroadcast=All in the connection string. See "Host (IP) protocol option" [*SQL Anywhere Server - Database Administration*] and "DoBroadcast (DOBROAD) protocol option" [*SQL Anywhere Server - Database Administration*].

- **ServerPort (PORT) protocol option** In previous releases, the PORT protocol option indicated one or more port numbers on which the database server may be listening, but was considered a hint to the client library. When the client library sent out a broadcast, it would use the port numbers specified by the PORT protocol option, as well as the default port number, 2638. In this release, if the PORT option is specified, only the specified ports are used to find the database server. See "ServerPort (PORT) protocol option" [*SQL Anywhere Server - Database Administration*].

- **Idle connection parameter respected for shared memory connections** The Idle connection parameter specifies a connection's idle timeout period. Shared memory connections now respect this connection parameter. The default idle timeout for shared memory connections is zero (never idle timeout). See "Idle connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Database server name may not be the same as the name specified by the ServerName (Server) connection parameter** In previous releases, the value of the Name database property always matched the value specified in the ServerName (Server) connection parameter. However, if a client connects to a database and uses the new NodeType (NODE) connection parameter and the client is redirected to connect to a different database server, then the names do not match. See "NodeType (NODE) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Some operations are not supported when connecting with an alternate server name** In previous releases, if a client connected to a database using an alternate server name, they could create, stop, and drop other databases on the same database server. These operations are no longer supported when you connect using an alternate server name.

- **Database mirroring behavior changes and deprecated features** Using the -xp server option to define database mirroring options such as the name of the arbiter server, the authentication string, and the synchronization mode is deprecated. However, you must still specify **-xp on** if you want to use the database server in a mirroring system. See "-xp dbsrv12 database option" [*SQL Anywhere Server - Database Administration*].

  You can now define the database mirroring settings using the following SQL statements:

  - "CREATE MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER MIRROR SERVER statement" [*SQL Anywhere Server - SQL Reference*]
  - "SET MIRROR OPTION statement" [*SQL Anywhere Server - SQL Reference*]

  The following behavior changes to database mirroring have been introduced in this release:

  - In previous releases, the name of the state information file for a mirror server defaulted to a name based on the server name. You must now specify the name of the state information file.
  - In previous releases, web service requests directed to the mirror server were redirected to the primary server. In this release, web service requests are handled by the server that receives them.

  For information about the enhancements to database mirroring in this release, see "Database mirroring enhancements" on page 6.

- **Embedded SQL cursor behavior change** Embedded SQL cursors now default to READ ONLY. Explicit FOR READ ONLY or FOR UPDATE clauses must now be specified in the PREPARE statement and not the DECLARE statement. See:
  - "DECLARE CURSOR statement [ESQL] [SP]" [*SQL Anywhere Server - SQL Reference*]
  - "PREPARE statement [ESQL]" [*SQL Anywhere Server - SQL Reference*]
- **Cursors closed for CREATE OR REPLACE PROCEDURE statements** When you run a CREATE OR REPLACE PROCEDURE statement, any cursors that are open for a connection are closed. See "CREATE PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*].
- **Back quote identifier delimiter** Back quotes can now be used as an identifier delimiter. See "Identifiers" [*SQL Anywhere Server - SQL Reference*].
- **Changes in locking behavior** To maximize concurrency, the key and non-key portions of a row can now be locked independently. Non-key columns of a row can be updated without interfering with the insertion and deletion of foreign rows referencing that row. See "How locking works" [*SQL Anywhere Server - SQL Usage*].
- **ODBC driver behavior change** The ODBC function SQLTables can be used to get a list of all schemas (users) by calling the function with the SQL_ALL_SCHEMAS argument. In previous versions, the list of users returned by this function only included users that owned a table. In a newly initialized database, this excluded some users, and in particular the DBA user. Now the complete list of schemas is returned, including those that do not own a table.
- **divide_by_zero_error option** The divide_by_zero_error option was not supported in version 11. This option is supported in version 12. If you are using materialized views, you must set this option to On (the default) to create new materialized views. See "divide_by_zero_error option" [*SQL Anywhere Server - Database Administration*] and "Restrictions on materialized views" [*SQL Anywhere Server - SQL Usage*].
- **PrefetchBuffer (PBUF) connection parameter** In previous releases, the amount of memory for buffering rows specified by the PrefetchBuffer (PBUF) connection parameter was shared between all connections. In this release, the amount of memory specified by this connection parameter is available to each connection. See "PrefetchBuffer (PBUF) connection parameter" [*SQL Anywhere Server - Database Administration*].
- **External logins automatically removed for dropped users** When you remove a user from the database, any external logins for the user are now dropped automatically. In previous releases, you had to remove the external logins separately. See "Deleting users from the database" [*SQL Anywhere Server - Database Administration*].
- **The database cleaner and database validation can no longer operate at the same time** In previous releases, database validation and the database cleaner could run at the same time, and report errors because of concurrent access to database pages. Database validation and the database cleaner no longer simultaneously operate on the same database. Validation waits for the database cleaner to finish, and the database cleaner waits for validation to finish if the database cleaner is started by calling sa_clean_database. The database cleaner can operate simultaneously with table or index validation.

- **Data types changed for columns in sa_index_density system procedure** The data type of the density and skew columns has been changed from numeric(8,6) to double. See "sa_index_density system procedure" [*SQL Anywhere Server - SQL Reference*].

- **DATEDIFF function** In previous releases, the DATEDIFF function returned an INTEGER for date parts of hours and smaller. DATEDIFF now returns an a BIGINT for these date parts. See "DATEDIFF function [Date and time]" [*SQL Anywhere Server - SQL Reference*].

- **openxml system procedure** In previous releases, the openxml system procedure converted NCHAR data to the CHAR encoding and parsed the data in CHAR format. Now NCHAR data is parsed in the NCHAR encoding if there are NCHAR columns in the output.

  In previous releases, the XPath arguments in the WITH clause could only be literal strings. Now literal strings and variables are allowed. See "openxml system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_text_index_vocab system procedure** Attempting to call sa_text_index_vocab on an NCHAR text index now returns an error. Use the new sa_text_index_vocab_nchar system procedure instead. See "sa_text_index_vocab_nchar system procedure" [*SQL Anywhere Server - SQL Reference*].

  Additionally:

  - The tabowner parameter is now optional.
  - sa_text_index_vocab can now be used with a CALL statement.
  - sa_text_index_vocab can now be used in a statement within a procedure.
  - The parameter values can now be host variables or expressions.

- **Default collation changed for Mac OS X** In previous releases, in the absence of any environment variables on Mac OS X, the Initialization utility (dbinit) would use ISO_8859-1:1987 for its default CHAR character set and ISO1LATIN1 for its CHAR collation (the same behavior as Linux). It now chooses UTF-8 with the UTF8BIN collation. See "Recommended character sets and collations" [*SQL Anywhere Server - Database Administration*].

- **Reserved words** The following is a list of reserved words added to the database in SQL Anywhere version 12.0.0:

  - datetimeoffset
  - inner
  - openxml
  - spatial
  - treat

  The following is a list of reserved words removed from the database in SQL Anywhere version 12.0.0:

- index_lparen
- lock
- with_cube
- with_lparen
- syntax_error
- with_rollup

SQL Anywhere deprecated and discontinued features

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

- **Address Windowing Extensions (AWE) deprecated** The use of Address Windowing Extensions for 32-bit Windows is deprecated. If you need a large cache, it is recommended that you use the 64-bit version of the SQL Anywhere database server on a 64-bit operating system. See "-cw dbeng12/dbsrv12 server option (deprecated)" [*SQL Anywhere Server - Database Administration*].

- **CALL statement** Use of this statement to invoke a function is deprecated. If you have a function you want to call, consider using an assignment statement to invoke the function and assign its result to a variable. For example:

  ```
  DECLARE varname INT; SET varname=test( );
  ```

  See "CALL statement" [*SQL Anywhere Server - SQL Reference*].

- **STOP ENGINE statement** The STOP ENGINE statement is deprecated. Use the STOP SERVER statement instead. See "STOP SERVER statement" [*SQL Anywhere Server - SQL Reference*].

- **Windows 2000 support removed** As of version 12.0.0, SQL Anywhere is no longer supported on Windows 2000.

- **Rebuild utility removed** The Rebuild utility is not supported in this release for rebuilding SQL Anywhere databases. You can rebuild databases using the Unload utility (dbunload). See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

- **Unsupported database properties** The following properties have been removed in this release:

  - CheckpointLogBitmapPagesWritten database property
  - CheckpointLogBitmapSize database property
  - java_main_userid connection property
  - QueryRowsBufferFetch connection property
  - QueryRowsBufferFetch database property

- **JDBC-based server classes deprecated** Support for the following JDBC-based server classes has been deprecated:

  - asejdbc
  - iqjdbc
  - sajdbc

  Applications should be updated to use the ODBC-based server classes. See "Defining ODBC external servers" [*SQL Anywhere Server - SQL Usage*].

- **SQL Anywhere Explorer no longer supported** The SQL Anywhere Explorer and SQL Anywhere Toolbar for Visual Studio are no longer supported. Use Microsoft's Server Explorer instead.

- **Short int embedded SQL indicator variable deprecated** To allow for the future use of 32- and 64-bit lengths and indicators, the use of short int for embedded SQL indicator variables is deprecated. Use a_sql_len instead. See "Indicator variables" [*SQL Anywhere Server - Programming*].

- **EngineName (ENG) connection parameter deprecated** The EngineName (ENG) connection parameter is deprecated. You can use the ServerName (Server) connection parameter instead. The short form of the ServerName connection parameter has been changed from ENG to Server. See "ServerName (Server) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **iAnywhere JDBC driver deprecated** The Type 1 iAnywhere JDBC driver is deprecated. Use the Type 2 SQL Anywhere JDBC driver instead. See "New SQL Anywhere TYPE-2 JDBC driver" on page 23.

- **-gu all database server option deprecated** The value **all** for the -gu database server option is deprecated. See "-gu dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **SET OPTION statement** The ability to specify an identifier as an option value in a SET OPTION statement, rather than a string literal, has been deprecated.

- **Service utility (dbsvc utility)** The value Standalone for the -t option has been deprecated. Use -t Personal instead to create a service for the personal database server. See "Service utility (dbsvc) for Linux" [*SQL Anywhere Server - Database Administration*] and "Service utility (dbsvc) for Windows" [*SQL Anywhere Server - Database Administration*].

MobiLink new features

Following is a list of additions to MobiLink introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **Central administration of remote databases** Central administration of remote databases can be used to do the following:

- Centrally control when a remote database synchronizes with MobiLink. This was previously set on the client with dbmlsync option or hard-coded into an application.
- Push schema changes to remote databases.
- Help diagnose problems with specific remote databases or with the synchronization system in general.

For information about central administration of remote databases, see "Central administration of remote databases" [*MobiLink - Server Administration*].

- **New MobiLink Replay utility (mlreplay)** The MobiLink Replay utility is a tool used to replay MobiLink protocol that is recorded by the MobiLink server. See "MobiLink replay utility (mlreplay)" [*MobiLink - Server Administration*].
- **MobiLink 12 plug-in for Sybase Central has been redesigned** The MobiLink plug-in has been redesigned in version 12 to support central administration of remote databases. The two MobiLink modes, Model and Admin, have been combined in the new plug-in. Now, you can use the MobiLink plug-in to create synchronization projects that contain consolidated databases, groups, synchronization models, and remote tasks. Old synchronization models can be imported into synchronization projects. See "Central administration of remote databases" [*MobiLink - Server Administration*].
- **Map consolidated columns to special values** You can now map consolidated columns to special values (such as the ML user) instead of remote columns.
- **Download delete subsets now supported** You can now specify download delete subsets, which by default are the same as the download subset.
- **New tree view allows selection of columns and tables** You can now choose the columns as well as the tables when creating or updating a consolidated database, or choosing tables to be synchronized in an existing remote database. To facilitate this, there is now a handy tree view of tables and columns where you can place checkmarks next to the tables and columns you want to synchronize.
- **Support for spatial data types** Synchronization models now support spatial data types and the TIMESTAMP WITH TIME ZONE data type.
- **Shadow tables now have indexes** Indexes are now created for any shadow tables that are created by the synchronization model. This can speed up download_delete_cursor and download_cursor scripts that use shadow tables.
- **SQL Anywhere Monitor now supports MobiLink server farms and Relay Server farms** Now, you can use the SQL Anywhere Monitor to monitor MobiLink server farms and Relay Server farms as well as SQL Anywhere databases and MobiLink servers. See "SQL Anywhere Monitor for MobiLink" [*MobiLink - Server Administration*].
- **New MobiLink arbiter server utility for server farms** For server-initiated synchronization and QAnywhere, the MobiLink arbiter ensures that only a single MobiLink server in a server farm is running as the primary server. See "Architecture" [*MobiLink - Getting Started*].

- **Enhanced support for server farms** New remote ID locking logic is used to prevent redundant synchronizations from the same remote ID in MobiLink high-availability. The -ss option no longer needs to be set. An arbiter is required when using server-initiated synchronization or QAnywhere with a MobiLink server farm.

  Separately licensed component required
  Running the MobiLink server in a server farm is a feature of the MobiLink high availability option, which requires a separate license. See "Separately licensed components" [*SQL Anywhere 12 - Introduction*].

- **TLS cipher suite support has changed** MobiLink server and clients now support 256-bit AES cipher suites for both RSA and ECC. Also, support has been added for the RFC 4492 version of the ECC cipher suites.

- **Dynamic memory caching** With dynamic memory caching, there is increased use of the memory cache so a larger cache may be needed to prevent swapping. Overall memory use should still be about the same.

- **New integrated Outbound Enabler** Use the new OE protocol for the -x option for mlsrv12 to use an integrated Outbound Enabler instead of the stand-alone Outbound Enabler invoked with the rsoe command. See "-x mlsrv12 option" [*MobiLink - Server Administration*].

- **New Relay Server plug-in for Sybase Central** There is a new Relay Server plug-in for Sybase Central that enables you to configure backend farms and servers. The Relay Server plug-in is only supported on Windows and Linux. See "Relay Server plug-in for Sybase Central" [*Relay Server*].

- **Upload and download data script requirement** Use the ml_add_missing_dnld_scripts stored procedure to fix missing download_cursor and/or download_delete_cursor scripts.

SQL statements

Following is a list of SQL enhancements for MobiLink introduced in SQL Anywhere version 12.0.0.

- **New SYNCHRONIZE statement [MobiLink]** This new statement allows you to synchronize a SQL Anywhere remote database with a MobiLink server. See "SYNCHRONIZE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

- **New START SYNCHRONIZATION SCHEMA CHANGE statement [MobiLink]** This new statement allows you to start a MobiLink synchronization schema change in a SQL Anywhere remote database. See "START SYNCHRONIZATION SCHEMA CHANGE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

- **New STOP SYNCHRONIZATION SCHEMA CHANGE statement [MobiLink]** This new statement allows you to stop a MobiLink synchronization schema change in a SQL Anywhere remote database. See "STOP SYNCHRONIZATION SCHEMA CHANGE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

- **CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink] enhancement** You can now specify a script version and subscription name. See "CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

- **ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink] enhancement** You can use the new SET SCRIPT VERSION clause to specify the script version to use during synchronization. You can also use the RENAME clause to rename a subscription. See "ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

Consolidated databases

Following is a list of enhancements to consolidated database support for MobiLink introduced in SQL Anywhere version 12.0.0.

- **Newly supported consolidated databases** The following new consolidated databases are supported by MobiLink in version 12.0.0:

  - IBM DB2 LUW 9.7
  - SQL Anywhere 12

- **Synchronization of spatial data** Synchronization of spatial data types is now supported for the following consolidated databases: SQL Anywhere, Oracle, Microsoft SQL Server, IBM DB2 and MySQL. See "Synchronizing spatial data" [*MobiLink - Server Administration*].

- **Support for longer CHAR columns** The MobiLink server now supports synchronizing CHAR, VARCHAR, NCHAR and NVARCHAR remote database columns with byte lengths greater than 32767 bytes.

- **Support for synchronizing columns with the data type of TIMESTAMP WITH TIME ZONE** The MobiLink server now supports synchronizing remote database columns with the data type of TIMESTAMP WITH TIME ZONE.

iAnywhere Solutions 12 - Oracle ODBC driver

Following is a list of enhancements to the iAnywhere Solutions 12 - Oracle ODBC driver introduced in SQL Anywhere version 12.0.0.

- **VARRAY support in packaged procedures** The iAnywhere Solutions 12 - Oracle ODBC driver now supports the use of VARRAYs in packaged procedures.

MobiLink server

Following is a list of enhancements to the MobiLink server introduced in SQL Anywhere version 12.0.0.

New mlsrv12 features

- **Server name change to mlsrv12** The MobiLink server has changed from mlsrv11 to mlsrv12.

- **New -cmax option for mlsrv12** This new option is part of the dynamic cache sizing feature. It sets the maximum size for the server memory cache. See "-cmax mlsrv12 option" [*MobiLink - Server Administration*].

- **New -cinit option for mlsrv12** This new option is part of the dynamic cache sizing feature. It sets minimum size for the server memory cache. See "-cmin mlsrv12 option" [*MobiLink - Server Administration*].

- **New -cmin option for mlsrv12** This new option is part of the dynamic cache sizing feature. It sets the initial size for the server memory cache. See "-cinit mlsrv12 option" [*MobiLink - Server Administration*].

- **New protocol option for -x option for mlsrv12** The -x option for mlsrv12 now supports a new oe protocol option that allows you to use the new integrated outbound enabler when using the Relay Server. See "-x mlsrv12 option" [*MobiLink - Server Administration*].

- **New default for -sl java option for mlsrv12** By default, the MobiLink server now tries to load a server Java VM if one is present instead of a client VM. To override this new default, you can explicitly request a client VM by adding -client to your -sl java options. See "-sl java mlsrv12 option" [*MobiLink - Server Administration*].

- **New -ca option for mlsrv12** Provide the host name that runs the MobiLink arbiter when using multiple servers with QAnywhere or server-initiated synchronization without lightweight polling. See "-ca mlsrv12 option" [*MobiLink - Server Administration*].

- **New -ftru option for mlsrv12** This option has been added to enable file upload support in the MobiLink server. This option allows you to set the root directory for files to be uploaded with the mlfiletransfer utility. See "-ftru mlsrv12 option" [*MobiLink - Server Administration*].

- **New metrics for -ppv option for mlsrv12** A number of new metrics have been added to the -ppv option for mlsrv12. See "-ppv mlsrv12 option" [*MobiLink - Server Administration*].

- **New -rrp option for mlsrv12** The new -rrp option for mlsrv12 was added in conjunction with the new MobiLink Replay utility (mlreplay). The option causes the MobiLink server to run the mlreplay utility and replay all recorded sessions in the given directory when the server starts. See "-rrp mlsrv12 option" [*MobiLink - Server Administration*].

- **New -rp option for mlsrv12** The new -rp option for mlsrv12 was added in conjunction with the new MobiLink Replay utility (mlreplay). The option is used to specify the directory to which synchronizations are recorded for playback with the mlreplay utility. See "-rp mlsrv12 option" [*MobiLink - Server Administration*].

- **New -vR option for mlsrv12** Use the new -vR option for mlsrv12 to show the remote ID in each log message for synchronization. See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **New -vU option for mlsrv12** Use the new -vU option for mlsrv12 to show the MobiLink user name in each log message for synchronization. See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **New -vk option for mlsrv12** This new option is part of the dynamic cache sizing feature. This option prints a line to the log whenever the cache grows or shrinks. See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **Improvements to low memory performance in MobiLink server** There is improved behavior and robustness in low-memory conditions.

- **Two new conflict detection approaches** The upload_fetch, upload_fetch_column_conflict, upload_new_row_insert, and upload_old_row_insert scripts can be used to detect conflicts. See "Detecting conflicts" [*MobiLink - Server Administration*].

- **New ml_add_missing_dnld_scripts system procedure** The new system procedure can be used to add missing download_cursor and download_delete_cursor scripts. See "ml_add_missing_dnld_scripts system procedure" [*MobiLink - Server Administration*].

New MobiLink server API features

- **New TimestampWithTimeZone class added (Java)** This class provides methods that allow you to retrieve and specify the time zone offset of a Timestamp value. See "TimestampWithTimeZone class" [*MobiLink - Server Administration*].

- **New DateTimeWithTimeZone class added (.NET)** This class provides methods that allow you to retrieve and specify the time zone offset of a DateTime value. See "DateTimeWithTimeZone class" [*MobiLink - Server Administration*].

- **New SpatialUtilities classes added (.NET and Java)** This class provides methods that assist you when working with spatial data. See "SpatialUtilities class" [*MobiLink - Server Administration*] (.NET) and "SpatialUtilities class" [*MobiLink - Server Administration*]. (Java)

New MobiLink scripting features

- **New system parameters available for upload_fetch and upload_fetch_column_conflict scripts** The MobiLink server system parameters **remote_id** and **username** are now available for the upload_fetch and upload_fetch_column_conflict scripts.

- **New parameters are available for authenticate_file_transfer scripts** New parameters are available for authenticate_file_transfer scripts. See "authenticate_file_transfer connection event" [*MobiLink - Server Administration*].

- **New authenticate_file_upload connection event** A new connection event is available to support file uploads. See "authenticate_file_upload connection event" [*MobiLink - Server Administration*].

- **New generate_next_last_download_timestamp script** Use this script instead of modify_next_last_download_timestamp if you want to suppress MobiLink's default algorithm for determining the next last download time. See "generate_next_last_download_timestamp event" [*MobiLink - Server Administration*].

Server utilities

Following is a list of enhancements to the MobiLink server utilities introduced in SQL Anywhere version 12.0.0.

- **New mlarbiter utility** The new mlarbiter utility is used to start the MobiLink arbiter server. The arbiter is required when using server-initiated synchronization or QAnywhere with a MobiLink server farm. See "MobiLink arbiter server utility (mlarbiter)" [*MobiLink - Server Administration*].

MobiLink Monitor

Following is a list of enhancements to the MobiLink Monitor introduced in SQL Anywhere version 12.0.0.

- **New entries in the graph pane** The following entries have been added to the graph pane in the MobiLink Monitor: OE work queue, Notifier work queue, and Dynamic Cache work queue. See "Using the Utilization Graph" [*MobiLink - Server Administration*].

- **NewTrusted Certificate File option** A new **Trusted Certificate File** option and **Browse** button has been added to the **Connect To MobiLink Server** window.

MobiLink clients

Following is a list of enhancements to MobiLink clients introduced in SQL Anywhere version 12.0.0.

- **Dbmlsync has enhanced support for background synchronization** The database engine can now drop the dbmlsync connection to the remote database (and rollback uncommitted operations dbmlsync has) if another connection is waiting for access to any database resource that dbmlsync has locked. This allows the other connection to go forward without waiting for the synchronization to complete.

  Use these options for background synchronization:

  - "-bk dbmlsync option" [*MobiLink - Client Administration*]
  - "-bkr dbmlsync option" [*MobiLink - Client Administration*]
  - Background and BackgroundRetry options for "MobiLink synchronization profiles" [*MobiLink - Client Administration*]

- **dbmlsync now supports named subscriptions** The new -s option for dbmlsync allows you to specify the names of subscriptions to be synchronized. See "-s dbmlsync option" [*MobiLink - Client Administration*].

- **HTTP response buffering** Use the new network protocol option http_buffer_responses to completely stream HTTP packets from MobiLink into an intermediary buffer before being processed instead of processing the bytes as they are read off the wire. See "http_buffer_responses" [*MobiLink - Client Administration*].

- **Client-side certificates can now be used to authenticate MobiLink clients to third party servers and proxies** The identity and identity_password synchronization parameters have been added to provide support for this feature.

- **Synchronization no longer required to implement schema changes on remote databases (for SQL Anywhere clients only)** You can simplify schema changes on the remote database by using the START and STOP SYNCHRONIZATION SCHEMA CHANGE statements to delimit your schema change. See:

  - "START SYNCHRONIZATION SCHEMA CHANGE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]
  - "STOP SYNCHRONIZATION SCHEMA CHANGE statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]

  Use the CREATE or ALTER SYNCHRONIZATION SUBSCRIPTION statement to specify a script version for each synchronization subscription. See:

  - "CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]

- **New mlfileupload method for UltraLite** The MLFileUpload method has been added for UltraLite clients.

- **Changes to the dbmlsync options window** The following changes have been made to the dbmlsync options window:

  - **Retry on remote progress**, **Remote is behind**, **Remote is ahead**, **Drop conflicting connections**, **Site Script**, and **Cmdline help** have been removed.
  - The **Publication** option has been replaced with a **Subscription** option.

New dbmlsync options

- **New -s option for dbmlsync** Use the new -s option for dbmlsync to specify the names of the subscriptions to be synchronized. See "-s dbmlsync option" [*MobiLink - Client Administration*].

- **New -bk option for dbmlsync** This option enables background synchronization. See "-bk dbmlsync option" [*MobiLink - Client Administration*].

- **New -bkr option for dbmlsync** This option controls the behavior of dbmlsync after a background synchronization is interrupted. See "-bkr dbmlsync option" [*MobiLink - Client Administration*].

- **New -ci option for dbmlsync** This option is part of the new dynamic cache sizing feature. Use this option to set the initial size of the cache used by dbmlsync for synchronization data. See "-ci dbmlsync option" [*MobiLink - Client Administration*]

- **New -cl option for dbmlsync** This option is part of the new dynamic cache sizing feature. Use this option to set the minimum size to which the dbmlsync cache file will be reduced. See "-cl dbmlsync option" [*MobiLink - Client Administration*].

- **New -cm option for dbmlsync** This option is part of the new dynamic cache sizing feature. Use this option to set the maximum size limit for the dbmlsync cache file. See "-cm dbmlsync option" [*MobiLink - Client Administration*].

- **New BufferDownload extended option for dbmlsync** This option is part of the new dynamic cache sizing feature. The option specifies whether the entire download from the MobiLink server should be read into the cache before applying it to the remote database. See "BufferDownload (bd) extended option" [*MobiLink - Client Administration*].

New Dbmlsync C++ API objects

- **CancelSync method** This method allows MobiLink clients to cancel a synchronization. See "CancelSync method" [*MobiLink - Client Administration*].

- **DBSC_CancelRet enumeration** This enumeration indicates the result of a synchronization cancellation attempt. See "DBSC_CancelRet enumeration" [*MobiLink - Client Administration*].

- **DBSC_ERR_ACTIVE_SYNC_NOT_CANCELED member** This member indicates that the server could not cancel the synchronization request because the synchronization was active. See "DBSC_ErrorType enumeration" [*MobiLink - Client Administration*].

- **DBSC_ERR_DEAD_SERVER member** This member indicates that the dbmlsync server encountered an error while starting up and is shutting down. See "DBSC_ErrorType enumeration" [*MobiLink - Client Administration*].

- **"enable status" property** Three new events (DBSC_EVENTTYPE_ML_CONNECT, DBSC_EVENTTYPE_UPLOAD_COMMITTED, and DBSC_EVENTTYPE_DOWNLOAD_COMMITTED) are sent to the MobiLink client when this property is enabled. See "DBSC_EventType enumeration" [*MobiLink - Client Administration*], "SetProperty method" [*MobiLink - Client Administration*] and "GetProperty method" [*MobiLink - Client Administration*].

New Dbmlsync .NET API objects

- **CancelSync method** This method allows MobiLink clients to cancel a synchronization. See "CancelSync method" [*MobiLink - Client Administration*].

- **DBSC_CancelRet enumeration** This enumeration indicates the result of a synchronization cancellation attempt. See "DBSC_CancelRet enumeration" [*MobiLink - Client Administration*].

- **DBSC_ERR_ACTIVE_SYNC_NOT_CANCELED member** This member indicates if the server could not cancel the synchronization request if the synchronization was active. See "DBSC_ErrorType enumeration" [*MobiLink - Client Administration*].

- **DBSC_ERR_DEAD_SERVER member** This member indicates that the dbmlsync server encountered an error while starting up and is shutting down. See "DBSC_ErrorType enumeration" [*MobiLink - Client Administration*].

- **"enable status" property** Three new events (DBSC_EVENTTYPE_ML_CONNECT, DBSC_EVENTTYPE_UPLOAD_COMMITTED, and DBSC_EVENTTYPE_DOWNLOAD_COMMITTED) are sent to the MobiLink client when this

property is enabled. See "DBSC_EventType enumeration" [*MobiLink - Client Administration*], "SetProperty method" [*MobiLink - Client Administration*] and "GetProperty method" [*MobiLink - Client Administration*].

New client utilities features

- **Upload files using mlfiletransfer** You can now upload files using the mlfiletransfer utility. See "MobiLink File Transfer utility (mlfiletransfer)" [*MobiLink - Client Administration*].

- **New -i option for mlfiletransfer utility** The -i option was added to mlfiletransfer to disable resume functionality. See "MobiLink File Transfer utility (mlfiletransfer)" [*MobiLink - Client Administration*].

Relay Server

The following Relay Server features have been added in this release:

- **Support for SQL Anywhere Monitor** Relay Server resources can now be monitored using the SQL Anywhere Monitor. See "SQL Anywhere Monitor for MobiLink" [*MobiLink - Server Administration*].

- **Support for central administration** The Relay Server supports central administration via Sybase Central.

- **RSOE supported on Mac OS** The Relay Server Outbound Enabler is now supported on Mac OS. For more detailed information about supported platforms, see http://www.sybase.com/detail?id=1061806.

- **Windows IIS 7 and 7.5 now supported** The Relay Server is now supported for IIS 7 on Windows 2008 and IIS 7.5 on Windows 2008 R2. For more detailed information about supported platforms, see http://www.sybase.com/detail?id=1061806.

- **New active_cookie option** The active_cookie option has been added to the backend farm section of the Relay Server configuration. See "Backend farm section properties" [*Relay Server*].

- **New active_header option** The active_header option has been added to the backend farm section of the Relay Server configuration. See "Backend farm section properties" [*Relay Server*].

- **Improvements to Relay Server troubleshooting** To improve troubleshooting, the Relay Server now has standardized Relay Server error codes with localized error messages and selective error messages with system error codes and embedded system error messages. See "Relay Server error messages" [*Error Messages*].

- **Support for SQL Anywhere database as a backend HTTP server** The Relay Server now supports SQL Anywhere failover and read-only scale out. See "SQL Anywhere web services high availability and scale-out solutions" [*SQL Anywhere Server - Database Administration*].

- **Outbound Enabler dynamic response buffer sizing** Dynamic response buffer sizing has significantly reduced the Outbound Enabler memory overhead.

- **Outbound Enabler user interface improvements** An instance identifier in window title, systray tip text and systray menu have all been to the Outbound Enabler user interface, and the Outbound Enabler status has been added to the systray tip text.

- **More efficient use of shared memory** The Relay Server uses shared memory (set by the shared_mem configuration option) more efficiently. In deployments with relatively slow reading clients exercising HTTP requests with large responses, the relay server is able to run with significantly less shared memory.

- **Relay Server Outbound Enabler now supports periodic backend server status requests using HTTP** The RSOE has been enhanced to support periodic backend server status requests using HTTP. A backend server status request is an alternative to the liveness ping and can be used to determine if the backend server is able to accept client requests or not. The new status_url parameter, which is specified as part of the rsoe -cs option, is used to enable periodic backend server status requests.

MobiLink behavior changes

Following is a list of behavior changes to MobiLink introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

MobiLink server changes

- **MobiLink server no longer requires *log4j.jar*** The *log4j.jar* file is no longer required by the MobiLink server and is no longer deployed with the MobiLink server. If you require *log4j.jar* you must install your own version of the jar and put it in the classpath.

- **New behavior for -cn option** The -cn option for mlsrv12 sets the maximum number of database connections used for database worker threads. In versions prior to SQL Anywhere 12, the -cn option for mlsrv12 set the maximum number of database connections. See "-cn mlsrv12 option" [*MobiLink - Server Administration*].

- **New behavior for -sl dnet option** Previously, the MobiLink server loaded the workstation CLR by default when using .NET scripts. It now loads the server CLR instead. You can restore the old behavior by adding **-clrFlavor=wks** to the -sl dnet option for mlsrv12. See "-sl dnet mlsrv12 option" [*MobiLink - Server Administration*].

- **MobiLink server now uses the GV_$TRANSACTION Oracle system view instead of V_$TRANSACTION** The Oracle account used by the MobiLink server must now have permission for the GV_$TRANSACTION Oracle system view instead of the V_$TRANSACTION system view. See "Oracle consolidated database" [*MobiLink - Server Administration*].

- **Upgrade scripts for version 6.0.x removed** The MobiLink upgrade scripts for 6.0.x have been removed. If you require this upgrade, contact Technical Support (http://www.sybase.com/support).

- **ml_add_column system procedure no longer required** For version 12 clients, the ml_add_column system procedure is no longer required when you want to use column names in

named parameters. By default, you can now reference the column names directly without any extra setup. See "ml_add_column system procedure" [*MobiLink - Server Administration*].

- **New BIGINT data type support for Java and .NET server APIs** The BIGINT SQL data type now maps to the LONG Java and .NET data types. See "SQL-Java data types" [*MobiLink - Server Administration*] and "SQL-.NET data types" [*MobiLink - Server Administration*].

- **Data scripts are now required** To reduce the chance of losing data from accidentally not creating data scripts, the MobiLink server now requires either an ignored script or a valid script for the following events. See "Ignoring scripts" [*MobiLink - Server Administration*].

  - **upload_insert** if any inserted rows are uploaded from remotes and no handle_UploadData connection script is defined.

  - **upload_update** if any updated rows are uploaded from remotes and no handle_UploadData connection script is defined.

  - **upload_delete** if any deleted rows are uploaded from remotes and no handle_UploadData connection script is defined.

  - **download_cursor and download_delete_cursor** if no handle_DownloadData connection script is defined and the synchronization is not upload-only.

  As a convenience when upgrading to version 12, you can use the ml_add_missing_dnld_scripts stored procedure to add ignored scripts to avoid errors from missing download scripts. See "ml_add_missing_dnld_scripts system procedure" [*MobiLink - Server Administration*].

MobiLink client changes

Following is a list of behavior changes to MobiLink clients introduced in version 12.0.0.

- **Version 12 MobiLink synchronization clients send column names by default** In version 12, MobiLink synchronization clients all send column names to the MobiLink server by default. This means that in most cases, you are no longer required to use the ml_add_column system stored procedure to define column names and ordering for use with named parameters in MobiLink scripts. See "ml_add_column system procedure" [*MobiLink - Server Administration*] and "SendColumnNames (scn) extended option" [*MobiLink - Client Administration*].

- **Added support for enhanced TLS session renegotiation** MobiLink clients now support a new TLS extension that allows vulnerable third-party servers to be secured.

- **MLFileTransfer method renamed** The MLFileTransfer method for UltraLite clients has been split into the MLFileUpload and MLFileDownload methods.

- **-df has been renamed to -lf** The -df option for the mlfiletransfer utility has been renamed to -lf and now refers to a local file instead of a destination file.

- **-dp has been renamed to -lp** The -dp option for the mlfiletransfer utility has been renamed to -lp and now refers to a local path instead of a destination path.

- **UPLD_ERR_USERID_ALREADY_IN_USE has been changed** Dbmlsync no longer returns UPLD_ERR_USERID_ALREADY_IN_USE as a failure cause for the sp_hook_dbmlsync_upload_end event hook. In its place, the value UPLD_ERR_REMOTE_ID_ALREADY_IN_USE is returned.

- **Palm no longer supported for UltraLite clients** Palm OS is no longer supported. If you wish to use Palm, you should continue to use SQL Anywhere 11.

MobiLink plug-in for Sybase Central changes

Following is a list of behavior changes for the MobiLink plug-in for Sybase Central introduced in version 12.0.0.

- **Changes to Mapping editors for Synchronization models**

  - The Table and Column Mapping editors for synchronization models now show consolidated databases on the left and remote databases on the right. See "Modifying table and column mappings" [*MobiLink - Getting Started*].

  - Only the synchronized consolidated tables are listed in the table mapping editor. Changing the table mapping direction to **Not Synchronized** is the same as deleting the table mapping, removing that row from the editor when the model is saved, without changing database schema.

  - To add a table mapping for an unsynchronized consolidated table, and optionally add the table to the remote schema, use the **New Table Mappings** command. You can also use the **Update Schema** command to change the model's consolidated or remote schema.

  - Popup menus are now used instead of dropdown lists for choosing the right-hand side of table and column mappings, with an optional dialog (accessed with the ellipsis button) if all the choices do not fit in the menu. For table mappings, only unsynchronized remote tables are listed. For column mappings, unsynchronized columns in the remote table are listed along with the options for value mappings and for not synchronizing the column.

  - If you change a synchronization option for a table mapping, the lower pane automatically switches to the tab for the corresponding sub-options.

- **VARBIT and LONG VARBIT columns** VARCHAR and LONG VARCHAR columns are used in place of VARBIT and LONG VARBIT columns, respectively, when deploying a synchronization model with a remote schema to a new UltraLite remote database.

- **GO now used as the statement delimiter** For SQL Anywhere and UltraLite databases, synchronization models now use GO for the statement delimiter instead of a semicolon, allowing the SQL generated with synchronization models to be used in central administration of remote tasks.

MobiLink deprecated and discontinued features

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

- **IBM DB2 Mainframe not supported in Version 12.0.0** IBM DB2 mainframe is not supported as a consolidated database in version 12.0.0. However, MobiLink still supports DB2 LUW (Linux, Unix, and Windows) as a consolidated database.
- **Adaptive Server Enterprise 12.5.x no longer supported** Adaptive Server Enterprise 12.5.x is no longer supported by MobiLink in version 12.0.0.
- **IBM DB2 LUW 8.2 no longer supported** IBM DB2 LUW 8.2 is no longer supported by MobiLink in version 12.0.0.
- **-xo option for mlsrv12 has been removed** Clients earlier than version 10 are no longer supported.
- **-f option for mlsrv12 has been removed** Use the -zf mlsrv12 option to specify that the MobiLink server should check for script changes at the beginning of each synchronization. See "-zf mlsrv12 option" [*MobiLink - Server Administration*].
- **-nba option for mlsrv12 has been removed** Blocking download acknowledgement is no longer supported. If the remote database requests a download acknowledgement, the MobiLink server will always use a non-blocking acknowledgement.
- **-fr option for mlsrv12 has been removed** The -fr option for mlsrv12 is no longer supported. If you want to ignore a script (which might cause data to be lost) then define the script as --{ml_ignore}.
- **Java and .NET data scripts returning SQL is deprecated** The ability for Java and .NET scripting logic to return strings that are interpreted by MobiLink server as SQL scripts is deprecated in data scripts. If your scripts need to cause changes in the consolidated database, they should do so directly from Java or .NET.

  See:

  - "Java and .NET data scripts returning SQL (deprecated)" [*MobiLink - Server Administration*]
  - "Direct row handling" [*MobiLink - Server Administration*]
  - "Data scripts" [*MobiLink - Server Administration*]
- **Download error hooks removed** The following dbmlsync hooks have been removed in version 12: sp_hook_dbmlsync_download_com_error, sp_hook_dbmlsync_download_fatal_sql_error, and sp_hook_dbmlsync_download_sql_error.
- **-f option for MobiLink File Transfer utility has been removed** The -f option for the mlfiletransfer utility is no longer supported.

- **-r option for MobiLink File Transfer utility has been removed** The -r option for the mlfiletransfer utility is no longer supported.

- **Memory (mem) and DownloadBufferSize (dbs) extended options for dbmlsync have been removed** The Memory (mem) and DownloadBufferSize (dbs) extended options for dbmlsync is are no longer supported. Use the CacheMin, CacheInit and CacheMax options instead.

- **dbmlsync support for SQL passthrough has been removed** The SQL passthrough feature is no longer supported for MobiLink clients. It has been replaced by the central administration of remote databases feature. See "Central administration of remote databases" [*MobiLink - Server Administration*].

- **-ss option for mlsrv12 no longer required** Prior to version 12, the -ss option for mlsrv12 was used enable the MobiLink server to run in a server farm environment. New remote ID locking logic that prevents redundant synchronizations, the -ss option is no longer necessary for MobiLink servers running in a server farm, and the option has been removed. An arbiter is required when using server-initiated synchronization or QAnywhere with a MobiLink server farm.

  Separately licensed component required
  Running the MobiLink server in a server farm is a feature of the MobiLink high availability option, which requires a separate license. See "Separately licensed components" [*SQL Anywhere 12 - Introduction*].

- **The MobiLink Redirector has been removed** The Redirector is no longer available. Use the Relay Server instead. See "Introduction to the Relay Server" [*Relay Server*].

- **Recommendation for using script versions** It is highly recommended that you no longer use the ScriptVersion extended option. Instead, use the SCRIPT VERSION clause on the CREATE SYNCHRONIZATION SUBSCRIPTION and ALTER SYNCHRONIZATION SUBSCRIPTION statements. See "CREATE SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*] and "ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*].

- **The -n option for dbmlsync has been deprecated** This option has been deprecated. It is recommended that you use the -s dbmlsync option instead. See "-s dbmlsync option" [*MobiLink - Client Administration*].

- **The -u option for dbmlsync has been deprecated** This option has been deprecated. It is recommended that you use the -s dbmlsync option instead. See "-s dbmlsync option" [*MobiLink - Client Administration*].

- **The Publication synchronization profile option has been deprecated** This option has been deprecated. It is recommended that you use the -s dbmlsync option instead. See "-s dbmlsync option" [*MobiLink - Client Administration*].

- **The MLUser synchronization profile option has been deprecated** This option has been deprecated. It is recommended that you use the -s dbmlsync option instead. See "-s dbmlsync option" [*MobiLink - Client Administration*].

- **The dbmlsync Integration component has been removed** The Dbmlsync integration component has been removed. In its place, use the dbmlsync programming interface. See "Dbmlsync API" [*MobiLink - Client Administration*].

- **Forced-conflict mode has been deprecated** The MobiLink server uses forced conflict resolution when the upload_insert, upload_update, and upload_delete script are all undefined. This feature has been deprecated.

- **Detecting conflicts with upload_update has been deprecated** You should either detect and resolve uploaded update conflicts in your upload_update script or use either an upload_fetch or upload_fetch_column_conflict script to detect conflicts. Relying on the MobiLink server to count affected rows by your upload_update script to detect a conflict, and then invoke your conflict resolution scripts, has been deprecated. See "Detecting conflicts with upload_update scripts (deprecated)" [*MobiLink - Server Administration*].

- **The use of question marks in SQL scripts** The use of plain question marks in MobiLink SQL scripts has been deprecated. Use named parameters instead. See "Named script parameters" [*MobiLink - Server Administration*].

QAnywhere new features

Following is a list of additions to QAnywhere introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **New light weight polling support** QAnywhere now has the ability to use light weight polling to receive push notifications from the MobiLink server. Use the following new options to use this new functionality:
  - The new lwpoll mode of the "-push qaagent option" [*QAnywhere*] and "-push qauagent option" [*QAnywhere*].
  - The new lwpoll property of the PUSH QAnywhere manager configuration property. See "QAnywhere manager configuration properties" [*QAnywhere*].

QAnywhere client API features

- **.NET QAManagerBase class supports ExceptionListener delegates** This support allows applications to be notified of QAExceptions that occur while receiving messages on the background thread. See "ExceptionListener delegate" [*QAnywhere*] and "SetExceptionListener method" [*QAnywhere*].

- **New ReOpen method added to QAManagerBase class (.NET and Java)** This method is called in an ExceptionListener or MessageListener to re-establish connections to the message store database. See "ReOpen method" [*QAnywhere*] (.NET) and "reOpen method" [*QAnywhere*]. (Java)

QAnywhere Agent features

- **-cd option for QAnywhere Agent** Use this option in conjunction with the -cr option to specify the delay between attempts to connect to the database. This option is valid for qaagent and qauagent. See "-cd qaagent option" [*QAnywhere*].

- **-cr option for QAnywhere Agent** Use this option to specify the number of times that the QAnywhere Agent should retry a failed connection to the database. This option is valid for qaagent and qauagent. See "-cr qaagent option" [*QAnywhere*].

QAnywhere behavior changes and deprecated features

Following is a list of deprecated features and behavior changes to QAnywhere introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

SQL Remote new features

Following is a list of additions to SQL Remote introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **SQL Remote now supports replication of spatial values** SQL Remote now supports replication of spatial data types. In addition, SQL Remote supports the replication of the new TIMESTAMP WITH TIMEZONE data type. See "TIMESTAMP WITH TIME ZONE data type" [*SQL Anywhere Server - SQL Reference*].

- **-g option added to Extraction utility (dbxtract)** By default, materialized views defined as MANUAL REFRESH are not initialized as part of a reload. You can now use the -g option with dbxtract to initialize these materialized views as part of the reload process. See "Extraction utility (dbxtract)" [*SQL Remote*].

- **SQL Remote now prints an error message when no publisher is defined** When SQL Remote connects to a database that has remote or consolidated users defined, but no publisher defined, then SQL Remote returns an error indicating that no publisher is defined in the database.

UltraLite new features

Following is a list of additions to UltraLite introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

General features

- **Support for spatial data** The following features have been added in support of the new spatial data capabilities in UltraLite:
  - **ST_Geometry data type** The ST_Geometry data type supports functions that can be applied to spatial values. See "ST_Geometry type" [*UltraLite - Database Management and Reference*].
  - **Spatial data functions** UltraLite provides several functions to support the processing of spatial data. See "Functions for spatial data" [*UltraLite - Database Management and Reference*].
- **New encryption examples** New code samples have been added to illustrate UltraLiteJ encryption on BlackBerry devices. For more information, see the *samples-dir\UltraLiteJ* directory.

Platforms and devices

iPhone and Mac OS X support

UltraLite applications can now be developed on Mac OS X to target Mac OS X and iPhone using the UltraLite C/C++ API. For more information, see "Developing applications using the UltraLite C++ API" [*UltraLite - C and C++ Programming*].There is also a detailed tutorial to assist you in developing an iPhone application. See "Tutorial: Build an iPhone application using the C++ API" [*UltraLite - C and C++ Programming*].

64-bit Linux machine installs

If you install SQL Anywhere on a 64-bit Linux machine you will also need to install the 32-bit subsystem separately. This is because some 64-bit Linux operating systems do not include pre-installed 32-bit compatibility libraries. To use 32-bit client software, you may need to install 32-bit compatibility libraries for your Linux distribution. For example, on Ubuntu, you may need to run the following command:

```
sudo apt-get install ia32-libs
```

Security

- **UltraLite supports FIPS 140-2 certified encryption** FIPS 140-2 certified encryption is now supported for UltraLite on 64-bit Windows. See "Deploy UltraLite with AES_FIPS database encryption" [*UltraLite - Database Management and Reference*].
- **UltraLite database encryption now uses 256-bit AES** UltraLite database encryption is now performed using 256-bit AES instead of 128-bit AES. See "Deploy UltraLite with AES_FIPS database encryption" [*UltraLite - Database Management and Reference*].

- **Salt is now used when hashing passwords** A random 4-byte salt value is now generated every time a new user is created or an existing user changes their password. See "UltraLite PWD connection parameter" [*UltraLite - Database Management and Reference*].

Utilities

- **UltraLite utilities improvements** The ulinit utility has been enhanced to allow you to create an UltraLite database based on information in a SQL Anywhere database even when the schema being extracted from the SQL Anywhere database contains elements that UltraLite does not support (such as column datatypes or default values for instance). This is now the default behavior (which can be turned off using the -f option). See "UltraLite Initialize Database utility (ulinit)" [*UltraLite - Database Management and Reference*]. The ulinit utility is no longer supported.

  Changes have also been made to the ulinit, ulerase, ulinfo, ulload, and ulsync utilities.

  See:

  - "UltraLite Initialize Database utility (ulinit)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Erase database (ulerase)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Information utility (ulinfo)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Load XML to Database utility (ulload)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Synchronization utility (ulsync)" [*UltraLite - Database Management and Reference*]

SQL

- **CREATE / DROP / ALTER USER added** See:

  - "CREATE USER statement [UltraLite]" [*UltraLite - Database Management and Reference*]
  - "DROP USER statement [UltraLite]" [*UltraLite - Database Management and Reference*]
  - "ALTER USER statement [UltraLite]" [*UltraLite - Database Management and Reference*]

- **New table constraint for CREATE TABLE and ALTER TABLE** An additional table constraint (SYNCHRONIZE ON | OFF | ALL) can be specified in a CREATE TABLE or an ALTER TABLE statement. See:

  - "CREATE TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "ALTER TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]

- **IF EXISTS clause added to DROP statements** A new IF EXISTS clause can now optionally be specified in DROP INDEX, DROP PUBLICATION, DROP SYNCHRONIZATION PROFILE, and DROP TABLE statements. See:
  - "DROP INDEX statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
- **IF NOT EXISTS clause added to CREATE statements** A new IF NOT EXISTS clause can now optionally be specified in CREATE TABLE, CREATE INDEX, CREATE PUBLICATION, and CREATE SYNCHRONIZATION PROFILE statements. See:
  - "CREATE TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "CREATE INDEX statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "CREATE PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
- **CREATE SYNCHRONIZATION PROFILE now also supports OR REPLACE clause** The CREATE SYNCHRONIZATION PROFILE statement now also support the option to replace a table. See "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].
- **COUNT_UPLOAD_ROWS function added** COUNT_UPLOAD_ROWS allows you to query the number of rows that will be uploaded in the next synchronization. See "COUNT_UPLOAD_ROWS function [Aggregate]" [*UltraLite - Database Management and Reference*].

Data types

- **ST_Geometry** The ST_Geometry data type supports functions that can be applied to spatial values. See "ST_Geometry type" [*UltraLite - Database Management and Reference*].
- **TIMESTAMP WITH TIMEZONE data type** The TIMESTAMP WITH TIMEZONE data type allows date and time values to be stored together with time zone offsets. A time zone offset is the number of minutes before or after UTC. See "UltraLite Initialize Database utility (ulinit)" [*UltraLite - Database Management and Reference*] and "UltraLite timestamp_with_time_zone_format creation parameter" [*UltraLite - Database Management and Reference*].

Programming interfaces

UltraLite C/C++

- **New UltraLite C/C++ API added** A new version of the UltraLite C/C++ API has been added to this release. This API is declared in the *ulcpp.h* header file. See "UltraLite C/C++ API reference" [*UltraLite - C and C++ Programming*].

  The following objects have been renamed for this version of the UltraLite C/C++ API:

  - ul_synch_info has been renamed to ul_sync_info.
  - ul_synch_result has been renamed to ul_sync_result.
  - ul_synch_status has been renamed to ul_sync_status.
  - ul_synch_stats has been renamed to ul_sync_stats.
  - ul_synch_observer_fn has been renamed to ul_sync_observer_fn.
  - ul_synch_state has been renamed to ul_sync_state.
  - ULInitSynchInfo has been renamed to ULInitSyncInfo.
  - ULSetSynchInfo has been renamed to ULSetSyncInfo.
  - ULGetSynchResult has been renamed to ULGetSyncResult.
  - All UL_SYNC_STATE objects have been renamed to UL_SYNC_STATE objects.
  - UL_SYNC_STATUS_FLAG_IS_BLOCKING has been renamed to UL_SYNC_STATUS_FLAG_IS_BLOCKING.
  - UL_SYNC_RESULT_ERROR_STRING_SIZE has been renamed to UL_SYNC_RESULT_ERROR_STRING_SIZE.
  - ULRegisterErrorCallback has been renamed to ULSetErrorCallback.
  - ULSetSynchronizationCallback has been renamed to ULSetSynchronizationCallback.

  The following methods have been removed from the C++ API: GetDatabaseID, SetDatabaseID, IsCaseSensitive, and GetCollationName. The functionality is now handled by GetDatabaseProperty and SetDatabaseOption. See "GetDatabaseProperty method" [*UltraLite - C and C++ Programming*] and "SetDatabaseOption method" [*UltraLite - C and C++ Programming*].

  The GetDatabasePropertyInt method has been added. See "GetDatabasePropertyInt method" [*UltraLite - C and C++ Programming*].

UltraLite.NET

- **New ULConnection.ValidateDatabase(ULDBValid) method added** This method has been added as an equivalent to calling ULConnection.ValidateDatabase(ULDBValid, String) while passing null for the tableName. See "ValidateDatabase(ULDBValid) method" [*UltraLite - .NET Programming*].

UltraLite for M-Business Anywhere

- **API features** The following objects have been added to this release:

  - The setPublications method in the SyncParms class. This method sets the publication to be synchronized. See "setPublications method" [*UltraLite - M-Business Anywhere Programming*].

- The getPublications method in the SyncParms class. This method returns the publication to be synchronized. See "getPublications method" [*UltraLite - M-Business Anywhere Programming*].
- The timestampWithTimeZoneFormat member in the CreationParms class. This member sets the format for the timestamp with the time zone retrieved from the database. See "timestampWithTimeZoneFormat variable" [*UltraLite - M-Business Anywhere Programming*].
- The setTimestampWithTimeZoneParameter method in the PreparedStatement class and the setTimestampWithTimeZone method in the ULTable class. These methods set the value for the specified SQLType.TIMESTAMP_WITH_TIME_ZONE type parameter using a Date object. See "setTimestampWithTimeZoneParameter method" [*UltraLite - M-Business Anywhere Programming*] and "setTimestampWithTimeZone method" [*UltraLite - M-Business Anywhere Programming*].
- The getTimestampWithTimeZone methods in the ResultSet and ULTable classes. These methods return the value for the specified column as a Date object. See "getTimestampWithTimeZone method" [*UltraLite - M-Business Anywhere Programming*] (ResultSet class) and "getTimestampWithTimeZone method" [*UltraLite - M-Business Anywhere Programming*]. (ULTable class).

UltraLite behavior changes and deprecated features

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

Following is a list of deprecated features and behavior changes to UltraLite introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

Deprecated features

- **ulcreate utility no longer supported** The ulcreate utility is no longer available. All of the functionality is now handled by ulinit. See "UltraLite Initialize Database utility (ulinit)" [*UltraLite - Database Management and Reference*]
- **SQL passthrough no longer supported** SQL passthrough, originally released with UltraLite 11, is no longer supported. This functionality is now handled by the "Central administration of remote databases" [*MobiLink - Server Administration*].
- **Option to ALTER SYNCHRONIZATION PROFILE using the OR REPLACE clause is no longer supported** The **OR REPLACE** clause has been removed from the **ALTER SYNCHRONIZATION PROFILE** statement.
- **UltraLite ODBC API no longer supported** The UltraLite ODBC API is no longer supported. Use the UltraLite C/C++ API in its place. See "UltraLite - C and C++ Programming".

Deprecated platforms

- **UltraLite for M-Business Anywhere no longer supported** UltraLite support for M-Business Anywhere is deprecated for UltraLite 12.

- **Palm operating system no longer supported** The Palm operating system is not supported by UltraLite 12.

Removed, deprecated, and modified APIs

- **Replaced UltraLite C/C++ API** The UltraLite C/C++ API defined in the *uliface.h* header file has been replaced by a new version that is defined in the *ulcpp.h* header file. The previous version of the API is still available. For documentation on the deprecated UltraLite C/C++ API, see http://dcx.sybase.com/1101en/ulc_en11/c-common-apiref.html.

  You can use the old implementation of the UltraLite C/C++ API by adding the *%SQLANY12%\SDK\ulcpp11.cpp* file to your UltraLite application project, where SQLANY12 is an environment variable that points to your SQL Anywhere installation directory.

- **Modified UltraLite C/C++ API objects** The following objects have been modified since the last release and apply to the new UltraLite C/C++ API:

  - SQL Passthrough is no longer supported by the API. The following objects have been removed:

    - ul_sql_passthrough_state
    - ul_sql_passthrough_status
    - ul_sql_passthrough_observer_fn

- **Modified UltraLite C/C++ common API objects** The following objects have been modified since the last release:

  - The MLFileTransfer function has been renamed to MLFileDownload. See "MLFileDownload method" [*UltraLite - C and C++ Programming*].

  - The force_transfer field of ml_file_transfer_info structure has been removed.

  - The enable_resume field of ml_file_transfer_info now defaults to true instead of false. See "MLFileDownload method" [*UltraLite - C and C++ Programming*].

  - MLFileDownload supports the new remote_key field of ml_file_transfer_info which is passed to MobiLink server scripts to allow greater control of file transfers. See "MLFileDownload method" [*UltraLite - C and C++ Programming*].

- **embedded Visual C++ not supported as of UltraLite 11.0** Support for Visual Studio 2003 ended with UltraLite 11.0. Support for embedded Visual C++ was therefore moved into Visual Studio 2005.

- **Modified Embedded SQL API objects** The following objects have been modified since the last release:

- SQL Passthrough is no longer supported by the API. The following objects have been removed:
  - ULGetSQLPassthroughScriptCount
  - ULExecuteNextSQLPassthroughScript
  - ULExecuteSQLPassthroughScripts
  - ULSetSQLPassthroughCallback

- **Modified UltraLite for M-Business Anywhere API objects** The following objects have been modified since the last release:
  - The GetSQLPassthroughScriptCount, ExecuteNextSQLPassthroughScript, and ExecuteSQLPassthroughScripts methods in the Connection class have been removed.
  - The syntax of the CreateDatabase method in the DatabaseManager class has changed.

- **Modified UltraLite.NET API objects** The following objects have been modified since the last release:
  - The DatabaseManager property under the ULConnection class has been removed and is no longer required. ULDatabaseManager is no longer a singleton class; the methods are now static. See "ULDatabaseManager class" [*UltraLite - .NET Programming*].
  - The DatabaseOnCE property in the ULConnectionParms class has been renamed to DatabaseOnDevice. See "DatabaseOnDevice property" [*UltraLite - .NET Programming*].
  - The GetOptimalIndex method in the ULTableSchema class now returns the name of the optimal index. See "GetOptimalIndex method" [*UltraLite - .NET Programming*].
  - The CountUploadRows(String, UInt32) method in the ULConnection class has been removed. Use CountUploadRows(String, Int64) instead. See "CountUploadRows method" [*UltraLite - .NET Programming*].
  - SQL Passthrough is no longer supported by the API. The following objects have been removed:
    - ULConnection.GetSQLPassthroughScriptCount
    - ULConnection.ExecuteNextSQLPassthroughScript
    - ULConnection.ExecuteSQLPassthroughScripts
    - ULSqlPassthroughProgressListener
    - ULSqlProgressData
    - ULSqlProgressState
  - The ULPublicationSchema class and its methods have been removed, along with the GetPublicationSchema method in the ULDatabaseSchema class. The SYNC_ALL_DB and SYNC_ALL_PUBS fields have moved to the ULConnection class. See "SYNC_ALL_DB field" [*UltraLite - .NET Programming*] and "SYNC_ALL_PUBS field" [*UltraLite - .NET Programming*].

Miscellaneous

- **User publication limit increase** The maximum number of user publications has increased to 63.

- **Default encoding for UltraLite databases is now UTF-8 encoded** UltraLite databases are now UTF-8 encoded by default. See "UltraLite utf8_encoding creation parameter" [*UltraLite - Database Management and Reference*].

UltraLiteJ new features

Following is a list of additions to UltraLiteJ introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **Support for external BLOB files** UltraLiteJ, using BlackBerry OS 4.2 or Java SE, supports the partitioning of database files such that external files may now be used to store large BLOB values, with the files referenced using specific columns in the database. This is implemented as part of the CREATE TABLE SQL function.

  A sample usage scenario might include having a BlackBerry application use the smaller but faster persistent store to store an UltraLiteJ database while storing large BLOB values, such as pictures, in larger (but slower) flash drives or SD cards. An added benefit is that applications that capture pictures and store them in the database do not waste battery power and time copying the pictures into the database.

  See "CREATE TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **BlackBerry internal flash and SD cards support** UltraLiteJ can read and write UltraLiteJ databases to either internal flash memory or SD cards. See "ConfigFileME interface (BlackBerry only)" [*UltraLiteJ*].

- **Support for multiple versions of UltraLiteJ** In order to allow multiple versions of UltraLiteJ to co-exist in the BlackBerry and Java ME environment, UltraLiteJ JAR files, COD files, and the Java package name now include the major version number, as follows:

  - The JAR file is called *UltraLiteJ12.jar*
  - The COD files are called *UltraLiteJ12.cod*
  - All public classes are contained in package com.ianywhere.ultralitej12
  - The class Unsigned64 has been moved to package com.ianywhere.ultralitej12

- **Encryption and securing UltraLiteJ applications** A new sample demonstrating how to write a very secure UltraLiteJ application for BlackBerry devices has been added. See *Samples\UltraLite\UltraLiteJ\BlackBerryEncryption\ReadMe.html* for details. For a more complete discussion on BlackBerry security, please read the white paper "UltraLiteJ Security on BlackBerry Devices" at http://www.sybase.com/detail_list?id=9814.

- **new ST_Geometry data type** The ST_Geometry data type supports functions that can be applied to spatial values. See "ST_Geometry type" [*UltraLite - Database Management and Reference*].

- **TIMESTAMP WITH TIMEZONE data type** The TIMESTAMP WITH TIMEZONE data type allows date and time values to be stored together with time zone offsets. A time zone offset is the number of minutes before or after UTC. See "UltraLite Initialize Database utility (ulinit)" [*UltraLite -*

Database Management and Reference] and "UltraLite timestamp_with_time_zone_format creation parameter" [*UltraLite - Database Management and Reference*].

- **RAND SQL function support** UltraLiteJ supports the RAND SQL function. See "RAND function [Numeric]" [*UltraLite - Database Management and Reference*].

- **Support for CREATE SYNCHRONIZATION PROFILE, ALTER SYNCHRONIZATION PROFILE, DROP SYNCHRONIZATION PROFILE, and SYNCHRONIZE** These statements are intended to provide an alternative way to organize sync parameters and to launch synchronizations using SQL. The existing Connection.createSyncParm(), Connection.synchronize(SyncParm) and related APIs continue to work. See:

  - "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "ALTER SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "SYNCHRONIZE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]

- **New table constraint for CREATE TABLE and ALTER TABLE** An additional table constraint (SYNCHRONIZE ON | OFF | ALL) can be specified in a CREATE TABLE or an ALTER TABLE statement. See:

  - "CREATE TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "ALTER TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]

- **IF EXISTS clause added** A new IF EXISTS clause can now optionally be specified in DROP INDEX, DROP PUBLICATION, DROP SYNCHRONIZATION PROFILE, and DROP TABLE statements. See:

  - "DROP INDEX statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "DROP TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]

- **IF NOT EXISTS clause added** A new IF NOT EXISTS clause can now optionally be specified in CREATE TABLE, CREATE INDEX, CREATE PUBLICATION, and CREATE SYNCHRONIZATION PROFILE statements. See:
  - "CREATE TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "CREATE INDEX statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "CREATE PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]
  - "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*]

- **CREATE SYNCHRONIZATION PROFILE now also supports OR REPLACE clause** The CREATE SYNCHRONIZATION PROFILE statement now also support the option to replace a table. See "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **File transfer through MobiLink** UltraLiteJ can upload and download files in the remote database through the MobiLink server.

  The desktop version of UltraLiteJ can download any type of file from MobiLink to the local file system, or upload any type of file in the local file system to MobiLink.

  The BlackBerry OS 4.2 version of UltraLiteJ can download valid, unencrypted, non-obfuscated database files from MobiLink and store them in the object store, or upload these types of databases to MobiLink. It can download any type of file from MobiLink to the media card or flash storage and upload any type of file from flash and media cards to MobiLink.

  For more information about UltraLiteJ file transfers, see "FileTransfer interface" [*UltraLiteJ*].

- **UltraLiteJ Database Transfer utility can open and transfer databases on the BlackBerry's file system** UltraLiteJ automatically decides the location of the database (object store or file system) based on the database name. If the name starts with *file://* (case sensitive), then the utility will try to find the database in the file system; otherwise it will find the database in the object store.

- **BlackBerry install directory renamed** The install directory for BlackBerry device files has been renamed to use the minimum BlackBerry OS version. The *UltraLite\UltraLiteJ\Java MERIM11* directory is now *UltraLite\UltraLiteJ\BlackBerry4.2* to indicate files compatible with BlackBerry OS 4.2 and later.

- **Blobfile type support for ULjLoad and ULjUnload** The UltraLiteJ load and unload utilities support custom implementations of the blobfile type. For a sample UltraLiteJ database implementation of blobfile types, see "UltraLiteJ Database Unload utility (ULjUnload)" [*UltraLiteJ*] and "UltraLiteJ Database Load utility (ULjLoad)" [*UltraLiteJ*].

- **Improved row limiting algorithm** The row limiting algorithm has been improved to consider that rows from tables with many columns may use more resources than rows from tables with few columns.

- **System table changes** The following changes have been made to the UltraLiteJ system tables:
  - **column_default_value column in syscolumn system table supports the DEFAULT AUTOFILENAME default clause** This column can handle VARCHAR types that are specified with a column default value of DEFAULT AUTOFILENAME. See "syscolumn system table" [*UltraLiteJ*].
  - **filename_colid column added to syscolumn system table** This column stores the column id of the referenced file_name column in the schema definition; otherwise, this column is null. See "syscolumn system table" [*UltraLiteJ*].
  - **table_partition_size column added to systable system table** This column stores the defined partition size value. See "systable system table" [*UltraLiteJ*].
- **Encryption performance changes** The following improvements have been made to the EncryptionControl interface to improve performance in slow CPU environments:
  - UltraLiteJ now only encrypts data and system critical pages.
  - The decrypt method now accepts an additional parameter to specify the number of bytes that need to be decrypted. See "decrypt method" [*UltraLiteJ*].

  For more information about the EncryptionControl interface, see "EncryptionControl interface" [*UltraLiteJ*].

New UltraLiteJ API features

- **API features** The following objects have been added to this release:
  - The ConfigFileMe interface has been added to provide methods that allow configurations for persistent databases saved in a file on a Java ME device file system. See "ConfigFileME interface (BlackBerry only)" [*UltraLiteJ*].
  - The setRowScoreFlushSize and setRowScoreMaximum methods in the ConfigPersistent interface have been added to replace the setRowMinimumThreshold and setRowMaximumThreshold methods. These methods accommodate the new row limiting algorithm. The row limiting algorithm improves the resource management of tables with many columns. See "setRowScoreFlushSize method" [*UltraLiteJ*] and "setRowScoreMaximum method" [*UltraLiteJ*].

    The UUIDValue interface has been added to describe a unique identifier. See "UUIDValue interface" [*UltraLiteJ*].
  - The getLastIdentity method in the Connection interface has been added to retrieve the most recent value inserted into an DEFAULT AUTOINCREMENT or DEFAULT GLOBAL AUTOINCREMENT column through the current connection. See "getLastIdentity method" [*UltraLiteJ*].
  - The getSyncObserver method in the Connection interface has been added to return the currently registered SyncObserver object for the connection. See "getSyncObserver method" [*UltraLiteJ*].

- The getSyncResult method in the Connection interface has been added to return the result of the last SYNCHRONIZE SQL statement executed on the connection. See "getSyncResult method" [*UltraLiteJ*].
- The setSyncObserver method in the Connection interface has been added to set a SyncObserver object to monitor the progress of synchronizations on the connection. See "setSyncObserver method" [*UltraLiteJ*].
- The createFileTransfer and createObjectStoreTransfer methods in the DatabaseManager interface have been added to create FileTransfer objects that can be used to transfer database files. See "createFileTransfer method" [*UltraLiteJ*] and "createObjectStoreTransfer method (Java ME BlackBerry only)" [*UltraLiteJ*].
- The FileTransfer interface has been added to provide methods that specify the options for a file transfer. See "FileTransfer interface" [*UltraLiteJ*].
- The FileTransferProgressData interface has been added to allow data to be passed to the observer callback. See "FileTransferProgressData interface" [*UltraLiteJ*].
- The FileTransferProgressListener interface has been added as the file transfer observer. See "FileTransferProgressListener interface" [*UltraLiteJ*].
- The getPlanTree method in the PreparedStatement interface has been added to present the plan for a query in a more readable fashion when displayed on a computer monitor or when printed. See "getPlanTree method" [*UltraLiteJ*].
- The hasShadowPaging method in the ConfigPersistent interface has been added to detect if shadow paging has been turned on. See "hasShadowPaging method" [*UltraLiteJ*].

UltraLiteJ behavior changes and deprecated features

Following is a list of deprecated features and behavior changes to UltraLiteJ introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

Removed, deprecated, and modified utility options

- **-p option for ULjInfo, ULjLoad and ULjUnload is now optional** **dba** is used as the default password when the **-p** option is not specified.

Deprecated platforms

- **UltraLiteJ no longer supports BlackBerry OS 4.1** UltraLiteJ now supports BlackBerry OS 4.2 or later.

Removed, deprecated, and modified APIs

- **Modified UltraLiteJ API objects** The following objects have been modified since the last release:

- The ianywhere.ultralitej package name has changed. The package name now include the version number to allow multiple versions of UltraLiteJ to co-exist in the BlackBerry and Java ME environment, UltraLiteJ JAR files, COD files. See "UltraLiteJ API reference" [*UltraLiteJ*].

- The table_autoinc column in UltraLiteJ syscolumn system table has been removed. See "syscolumn system table" [*UltraLiteJ*].

- The CollectionOfValueReaders interface has been removed. All CollectionOfValueReader methods have been moved to the ResultSet interface. See "ResultSet interface" [*UltraLiteJ*].

- The CollectionOfValueWriters interface has been removed. All CollectionOfValueWriter methods have been moved to the PreparedStatement interface. See "PreparedStatement interface" [*UltraLiteJ*].

- The ConfigPersistent methods setRowMaximumThreshold and setRowMinimumThreshold have been replaced by setRowScoreFlushSize and setRowScoreMaximum. See "setRowScoreFlushSize method" [*UltraLiteJ*] and "setRowScoreMaximum method" [*UltraLiteJ*].

- All schema-related methods and interfaces have been removed. The ForeignKeySchema interface and all methods in the ColumnSchema, IndexSchema, and TableSchema interfaces have been removed. The following methods have been removed from the Connection interface:

  - createPublication
  - createTable
  - dropForeignKey
  - dropPublication
  - dropTable
  - enableSynchronization
  - renameTable
  - schemaCreateBegin
  - schemaCreateComplete
  - setNeverSynchronized
  - startSynchronizationDelete
  - stopSynchronizationDelete
  - truncateTable

  Use UltraLiteJ supported SQL statements, such as CREATE TABLE and CREATE PUBLICATION, to perform schema operations. See "UltraLite SQL statements" [*UltraLite - Database Management and Reference*].

- The setDatabaseId and getDatabasePartitionSize methods of the Connection interface have been removed. A default partition size can no longer be specified. Use the DEFAULT GLOBAL AUTOINCREMENT statement to override the default partition size. See "Using GLOBAL AUTOINCREMENT in UltraLite" [*UltraLite - Database Management and Reference*].

- The getDatabaseID method of the Connection interface now has the same effect as calling Connection.getOption(OPTION_DATABASE_ID). See "getOption method" [*UltraLiteJ*].

- The Value, ValueReader, and ValueWriter classes have been removed.

Miscellaneous

- **User publication limit increase** The maximum number of user publications has increased to 63.

Administration tools new features

Following is a list of additions to administration tools introduced in version 12.0.0. For information about supported platforms and versions, see http://www.sybase.com/detail?id=1061806.

- **Connect window** The layout of the **Connect** window has been streamlined in version 12.0.0 and the **Connect Assistant** has been removed. Previously, you had to know which options were required for the type of connection you wanted to make. Now, you choose the connection type and the **Connect** window presents you with the options that are applicable to your specified connection type.

  Note

  - You must choose the one of the following connection types from the **Action** dropdown list:

    Connect To A Running Database On This Computer Connects to a database that is already running on your computer.

    Connect With An ODBC Data Source Connect to a database using an ODBC data source.

    Connect To A Running Database On Another Computer Connects to a database that is running on another computer in the network.

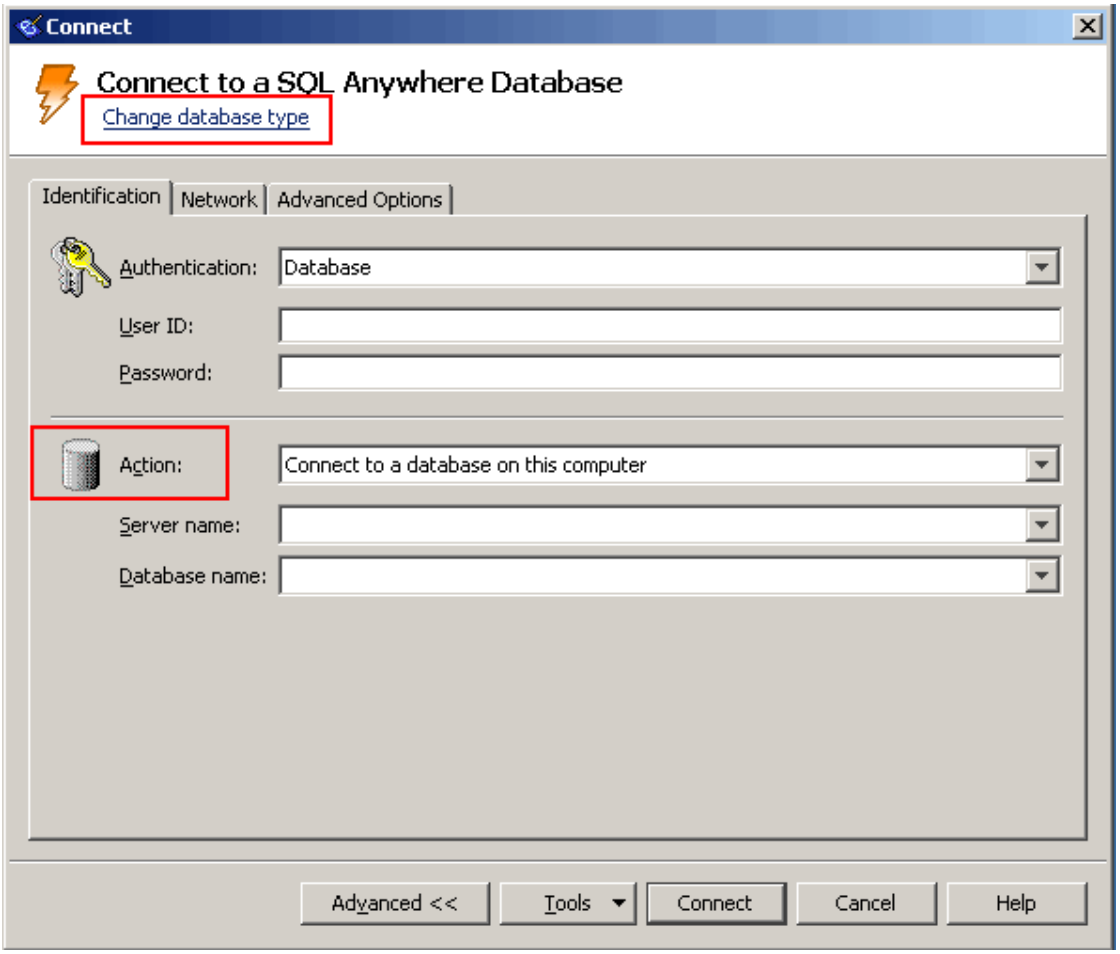
    Start And Connect To A Database On This Computer Starts a database on this computer and connects to it.

    Start And Connect To A Database On Another Computer Starts a database over a network on another computer and connects to it.

    Connect With A Connection String Connects to a database using a connection string.

    The options below the **Action** dropdown list change depending upon your choice.

  - In Interactive SQL, underneath the **Connect To A SQL Anywhere Database** heading, click **Change Database Type** to change the type of database that you are connecting to.



For example, when you are connecting to a database with an ODBC data source, the **Connect** window displays only two options: **ODBC Data Source Name** and **ODBC Data Source File**.

If required, you can click **Advanced** to specify options such as TCP/IP, encryption, and other advanced options. See "Working with the Connect window" [*SQL Anywhere Server - Database Administration*].

- **New default for checking software updates** Now, Interactive SQL, Sybase Central, SQL Anywhere Console utility, and MobiLink Monitor check for updates daily by default. In previous versions of the software, the default was to never check. See "Checking for software updates" [*SQL Anywhere Server - Database Administration*].

- **Viewing images, HTML, and XML data in Interactive SQL and Sybase Central** You can preview images, HTML, and XML data in a result set. See "Viewing images in Interactive SQL" [*SQL Anywhere Server - Database Administration*] and "Viewing HTML and XML data in Interactive SQL" [*SQL Anywhere Server - Database Administration*].

- **New system tray icons for Sybase Central and Interactive SQL** When the Sybase Central or the Interactive SQL fast launcher option is enabled, an icon now appears in the system tray. Right-click the icon and choose **Open** to open the application or choose **Exit** to close the application (if it is running) and terminate the fast launcher process. See "Using the fast launcher option" [*SQL Anywhere Server - Database Administration*].

- **Accessibility Enablement option** The Accessibility Enablement option is now installed by default. Previously you had to install this option separately. The Accessibility Enablement option allows Interactive SQL, Sybase Central, the SQL Anywhere Console utility, and the MobiLink Monitor to work with accessibility aids such as screen readers. See "Accessibility Enablement option" [*SQL Anywhere 12 - Introduction*].

Sybase Central plug-in new features

Following is a list of additions to Sybase Central plug-ins introduced in version 12.0.0.

SQL Anywhere plug-in new features

- **View the SQL statements and utility commands created by wizards** Most SQL Anywhere plug-in wizards that create a database object or run a database utility include a new page at the end of the wizard. This page displays the SQL statements and utility commands that are executed when you click **Finish**. Clicking **Finish** executes the SQL statements and/or runs the utility commands.

  Alternatively, you can copy the SQL statement to the clipboard, click **Cancel** to exit the wizard, and then execute the SQL statements via Interactive SQL. This feature allows you to use the wizards to generate SQL scripts without modifying the database. See "Viewing SQL statements and utility commands generated by wizards" [*SQL Anywhere Server - Database Administration*].

- **Support for spatial data** For information about the new plug-in features, including wizards, related to spatial data, see "Support for spatial data" on page 3.

- **New database Fragmentation tab** You can view a graphical representation of the fragmentation of base tables and indexes. The **Fragmentation** tab provides a graphical representation of the results from running sa_table_fragmentation system procedure on base tables. See "Using the Fragmentation tab (SQL Anywhere plug-in)" [*SQL Anywhere Server - SQL Usage*].

- **Logging database changes** You can log all SQL statements executed by the SQL Anywhere plug-in that modify the database. The log is stored in a *.sql* file. See "Logging database changes" [*SQL Anywhere Server - Database Administration*].

- **Support for WITH NULLS NOT DISTINCT clause, CREATE INDEX statement** You can use the SQL Anywhere plug-in to, create, view, and alter indexes that use the WITH NULLS NOT DISTINCT clause. The Indexes folder contains a **Nulls Distinct** column. The value in the column is blank when the index is a primary key, foreign key, unique constraint, or a non-unique index.

  See the UNIQUE clause of the "CREATE INDEX statement" [*SQL Anywhere Server - SQL Reference*].

- **Term breakers** The SQL Anywhere plug-in supports external term breakers and prefilters for text configuration objects. See "ALTER TEXT CONFIGURATION statement" [*SQL Anywhere Server - SQL Reference*].

- **Improvements to database documentation** When you use the **Database Documentation Wizard**, the generated documentation contains information about tables. The table information includes the owner name, procedures that modify the table, column information, and comments. See "Documenting a database" [*SQL Anywhere Server - Database Administration*].

- **Improvements to the Configure Type Filter window** The **Configure Type Filter** window lets you specify which database objects appear in the folder list in the left pane of Sybase Central. You can set your specifications to be the default type filter. Sybase Central uses the default filter whenever it connects to a database that does not have a type filter specified. See "Configuring the left pane in Sybase Central" [*SQL Anywhere Server - Database Administration*].

- **Improvements to Create Maintenance Plan Wizard** Maintenance plans contain user-defined operations. In the **Create Maintenance Plan Wizard** you can add user-defined operations as SQL statements that run either before validation or after backup.

  In addition, the **Maintenance Plans** folder now includes a **Status** column. See "Creating a maintenance plan" [*SQL Anywhere Server - Database Administration*].

- **Improvements to the Create Function Wizard** Previously, the **Create Function Wizard** limited the return type to a built-in type. Now, you can choose either a built-in type or a domain within the wizard. See "Creating user-defined functions" [*SQL Anywhere Server - SQL Usage*].

- **Improvements to the Backup Database Wizard** By default the **Backup Database Wizard** now enables free page elimination. See "Use the Backup Database Wizard" [*SQL Anywhere Server - Database Administration*].

- **Overview tab enhancement** The database **Overview** tab shows information about the health and statistics of copy servers along with information about database mirroring. See "Monitoring database health and statistics" [*SQL Anywhere Server - Database Administration*].

- **Improvements to the Synchronization Subscription Properties window** For databases created with SQL Anywhere 12, the name of the synchronization subscription appears in the **Synchronization Subscription Properties** window, as well as on the **Synchronization Subscriptions** tab. For databases that are created with earlier versions of SQL Anywhere, the name appears as **(unnamed)**. See "Creating synchronization subscriptions" [*MobiLink - Client Administration*].

- **New Synchronize Using Synchronization Profile window** You can synchronize a SQL Anywhere database by right-clicking a synchronization profile, choosing **Synchronize**, and clicking **OK**. See "MobiLink synchronization profiles" [*MobiLink - Client Administration*].

Sybase Central behavior changes and deprecated features

Following is a list of changes to Sybase Central introduced in version 12.0.0.

- **Sybase Central version changes to 6.1.0** SQL Anywhere 12 includes version 6.1.0 of Sybase Central 6.1.0.

- **Support for 32- and 64-bit computers** The Sybase Central configuration file has been renamed:

  - On 32-bit computers, the *.scRepository600* file is now named *.scRepository610_32*

  - On 64-bit computers, the *.scRepository600* file can be named *.scRepository610_32* and/or *.scRepository610_64* depending upon your installation.

- **When deploying Sybase Central, you no longer need to create JPR files for the plug-ins** Previously, when deploying Sybase Central and SQL Anywhere you needed to create JPR files for each plug-in. Now, Sybase Central uses the environment variable *%SQLANY12%* to find and register the plug-ins.

SQL Anywhere plug-in changed features

- **Maintenance plan enhancements** The following enhancements have been made to maintenance plans:

  - You can include SQL statements that are executed before validation
  - You can include SQL statements that are executed after backup
  - You can view the status of a maintenance plan while it is running
  - You can only run maintenance plan at a time

  For more information, see "Creating a maintenance plan" [*SQL Anywhere Server - Database Administration*].

- **Automatically refresh dynamic objects and properties** The lists of connections and locks, as well as the dynamic properties for events, maintenance plans, and windows services are automatically refreshed every 10 seconds by default. See "Set the refresh frequency" [*SQL Anywhere Server - Database Administration*].

- **New default for the Create Database Wizard** When you use the **Create Database Wizard** to create a new version 12 database, global checksums are now enabled by default. When you use the wizard to create a version 11 or later database, global checksums are disabled by default. Global checksums is always enabled by default when creating databases for Windows Mobile. See "Checksums enabled by default for new databases" on page 29.

Interactive SQL new features

Following is a list of additions to Interactive SQL introduced in version 12.0.0.

- **View spatial data using the Spatial Preview and Spatial Viewer** See "View spatial data as images" [*SQL Anywhere Server - Spatial Data Support*].

- **New ways to execute COMMIT and ROLLBACK statements** In Interactive SQL you can choose **SQL** » **Commit** to execute a COMMIT statement and you can choose **SQL** » **Rollback** to

execute a ROLLBACK statement. You can also use the keyboard shortcuts: Ctrl+Shift+C to execute a COMMIT statement and Ctrl+Shift+R to execute a ROLLBACK statement.

Executing a COMMIT or ROLLBACK via the **SQL** menu or a keyboard shortcut does not modify the contents of the **SQL Statements** pane; however, the **Results** tab in the **Results** pane is cleared. See "Executing COMMIT and ROLLBACK statements in Interactive SQL" [*SQL Anywhere Server - Database Administration*].

- **Changes to how you can select and copy columns, rows, and cells from result sets** In the Interactive SQL **Results** pane you can select multiple columns, rows, and cells in a result set, and then copy them. For example, to select multiple columns, hold the Ctrl key while clicking cells from the columns you want to copy, and then right-click and choose **Copy Data** » **Columns**. See "Copying columns, rows, and cells from an Interactive SQL result set" [*SQL Anywhere Server - Database Administration*].

- **Prevent OEM users from saving passwords in their favorites** OEM deployments can now prevent users from saving passwords in their connection favorites in Interactive SQL. See the allowPasswordsInFavorites option in "Configuring the administration tools" [*SQL Anywhere Server - Programming*].

- **Editing, importing, and exporting Favorites** Now you can edit, export, and import your Interactive SQL favorites. See "Using favorites" [*SQL Anywhere Server - Database Administration*] and "Sharing Favorites" [*SQL Anywhere Server - Database Administration*].

- **Execution times** The status bar in Interactive SQL shows the length of time that the current SQL statement has been executing.

- **Changes to the Results pane**

  - **Display results as text or in a scrolling table** Previously, you could only configure the display of a result set in the **Results** pane by changing settings in the **Options** window. Now, you can choose **Data** » **Show Results As Scrollable Table** to display the result set in a scrollable table. You can also choose **Data** » **Show Results As Text** to display the result set as text. You must execute a statement to see the change take effect. See "Customizing Interactive SQL" [*SQL Anywhere Server - Database Administration*].

  - **Sizing columns in Interactive SQL** You can right-click a result set and choose whether the columns should be sized to fit the window or to fit the data. See "Customizing Interactive SQL" [*SQL Anywhere Server - Database Administration*].

- **Help for SQL functions** In the **SQL Statements** pane, you can right-click a SQL function and choose **Help For *function-name*** and the documentation for the function appears.

- **Suppress warning messages in Interactive SQL** You can disable some of the warning messages that appear in Interactive SQL. For example you can suppress the warning that appears when you have unsaved text in the **SQL Statements** pane.

You can disable the warning message by choosing **Tools** » **Options** » **Messages** and then clearing the checkboxes in the **Optional Messages** list. See "Customizing Interactive SQL" [*SQL Anywhere Server - Database Administration*].

- **Recovering files in Interactive SQL** Interactive SQL attempts to recover unsaved changes to *.sql* files when Interactive SQL closes unexpectedly. When you edit a file, Interactive SQL makes a backup copy of the file 30 seconds after the last change and before you execute a statement.

Interactive SQL behavior changes and deprecated features

Following is a list of changes to Interactive SQL introduced in version 12.0.0.

- **Support for 32- and 64-bit computers**
  - Now, you can have both 32- and 64-bit versions of Interactive SQL installed on the same computer.
- **Changes to the default encoding for Windows** The following change applies to running Interactive SQL as a console application (with no windowed user interface) on Windows computers where the ANSI and OEM encodings differ, for example on a U.S. English Windows XP computer.

  1. Previously, when running Interactive SQL as a console application, the INPUT and READ statements assumed that the file was encoded using the OEM encoding (cp437 on a U.S. English Windows XP computer) in the absence of an explicit ENCODING clause. Similarly, the OUTPUT statement would output the file using the OEM encoding.

     Now, when running Interactive SQL as a console application, the INPUT and READ statements assume that the file is encoded using the ANSI encoding (cp1252 on U.S. English Windows XP computer). Similarly, the OUTPUT statement outputs the file using the ANSI encoding.

     From the command prompt, to process a file that uses the OEM encoding, you must specify the encoding explicitly. For example:

     ```
     dbisql READ ENCODING 'cp437' myfile.sql
     ```

  2. Previously, when running Interactive SQL as a console application, results written to and read from the command prompt used the ANSI encoding (cp1252 on a U.S. Windows XP computer), which could cause extended characters to be displayed incorrectly.

     Now, when running Interactive SQL as a console application, results written to and read from the command prompt use the OEM encoding (cp437 on a U.S. Windows XP computer).

  See "default_isql_encoding option [Interactive SQL]" [*SQL Anywhere Server - Database Administration*].

- **Change to the CLEAR statement, Clear menu item, and the Esc key** Now, the CLEAR statement closes any open result sets and leaves the contents of the **SQL Statements** pane unchanged. See "CLEAR statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

Also, the **Edit** » **Close Results** menu item, which is equivalent to executing a CLEAR statement, closes any open results sets and leaves the contents of the SQL Statements pane unchanged.

The **Edit** » **Clear** menu item which used to clear the contents of the SQL Statements pane has been removed. As a result, the keyboard shortcut for the **Clear** menu item, the Esc key, has also been removed. Now by default, pressing the Esc key has no affect.

However, you can set the Esc key to clear the **SQL Statements** pane and close any opened result sets. Choose **Tools** » **Options** » **Compatibility** and select **Pressing The Esc Key Clears SQL Statements And Closes Result Sets**.

- **-codepage option removed** If you want Interactive SQL to read a file with a specific code page, use the ENCODING clause of the INPUT, OUTPUT, or READ statements. The -codepage option has been removed from the software. See:
  - "Interactive SQL utility (dbisql)" [*SQL Anywhere Server - Database Administration*]
  - "Interactive SQL for UltraLite utility (dbisql)" [*UltraLite - Database Management and Reference*]
  - "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*]
  - "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*]
  - "READ statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*]
- **Change to SET OPTION statement [Interactive SQL]** Previously, if you used the SET OPTION statement to set an option and didn't specify a value, the option was set to Off. Now, if the option value is omitted, the specified option is set to its default value. This change affects the following options: auto_commit, auto_refetch, bell, commit_on_exit, and echo. See "SET OPTION statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].
- **Change to OUTPUT statement** When outputting results to a TEXT file, you can use the WITH COLUMN NAMES clause to insert the column names at the beginning of the file. See "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].
- **Change to INPUT statement** When inserting lines from a TEXT file with the INPUT statement, you can now use the SKIP clause to specify a number of lines to omit from the start of the file. See "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].
- **READ keyword no longer required** When you run Interactive SQL from the command prompt, the READ keyword is optional when specifying a *.sql* file to run. If the *.sql* file requires parameters, specify the parameters after the file name.

  For example, the following two commands are equivalent:

With the READ keyword	Without the READ keyword
`READ file.sql parm1`	`file.sql parm1`

  See "Interactive SQL utility (dbisql)" [*SQL Anywhere Server - Database Administration*].
- **Improvements to Microsoft Excel ODBC driver support** The following list describes changes related to exporting data from SQL Anywhere to Excel files via the Microsoft Excel ODBC driver:

- Previously you could not export data that was stored as a CHAR, LONG VARCHAR, NCHAR, NVARCHAR, or LONG NVARCHAR data types.

  Now when you export data that is stored as a CHAR, LONG VARCHAR, NCHAR, NVARCHAR, or LONG NVARCHAR data types from a SQL Anywhere database using the Microsoft Excel ODBC driver, the data is stored as VARCHAR (the closest type supported by the Excel driver).

  The Microsoft Excel ODBC driver supports text column widths up to 255 characters.

- You can export data that is stored as REAL, FLOAT, and BIGINT data types.

- Data stored as MONEY and SMALLMONEY data types are exported to the CURRENCY data type. Otherwise numerical data is exported as numbers.

- You can use the **Export Wizard** to export tables.

See "Exporting data" [*SQL Anywhere Server - SQL Usage*].

SQL Anywhere Monitor new features

Following is a list of additions to SQL Anywhere Monitor introduced in version 12.0.0.

- **New user interface** The Monitor user interface is dashboard-based. Dashboards contain widgets to display metrics, alerts, and resource information. Dashboards are specific to each user. Any user can add, edit, or delete their dashboards. See "Overview dashboard" [*SQL Anywhere Server - Database Administration*].

- **Monitor MobiLink server farms and Relay Server farms** You can use the Monitor to monitor MobiLink server farms and Relay Server Farms, as well as SQL Anywhere databases and MobiLink servers. See "SQL Anywhere Monitor for MobiLink" [*MobiLink - Server Administration*] and "SQL Anywhere Monitor for Relay Server" [*Relay Server*].

- **Close connections from within the Monitor** From within the Monitor, you can close connections to resource databases. See "Closing connections from the Monitor" [*SQL Anywhere Server - Database Administration*].

- **Import SQL Anywhere resources to be monitored** You can create a CSV file that contains a list of resources, and import this list into the Monitor. Previously, you could only add one resource at a time to the Monitor. See "Add multiple resources" [*SQL Anywhere Server - Database Administration*].

- **Perform on-demand maintenance** Administrators can perform unscheduled maintenance on the Monitor. See "Back up the Monitor" [*SQL Anywhere Server - Database Administration*].

- **New backup alert for SQL Anywhere resources** Administrators can configure the Monitor to issue an alert when a SQL Anywhere resource has not been successfully backed up for a given number of days. By default, the Monitor issues an alert after 14 days have past since a resource was

successfully backed up. See "Specify alert thresholds" [*SQL Anywhere Server - Database Administration*].

- **Time is reported local to the browser** The times reported in the Monitor are always local to the browser you are using. To calculate the difference between the browser time and the resource time, see "Understanding how time is displayed" [*SQL Anywhere Server - Database Administration*].

- **Export metrics** You can export metrics that have a graph or table associated with them to an XML file. For example, most of the metrics in the **Key Performance Metrics** widget can be exported. See "Export metrics" [*SQL Anywhere Server - Database Administration*].

- **Troubleshooting features** When you need to troubleshoot the Monitor, Administrators can use the **Message Log** and **Exception Reports** features. See "Message Log" [*SQL Anywhere Server - Database Administration*] and "Exception Reports" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere Monitor behavior changes

Following is a list of changes to SQL Anywhere Monitor introduced in version 12.0.0.

- **Changes to metric collection** Previously you could configure the type of metrics that the Monitor collected and set the alert thresholds. Now you can only configure the alert thresholds. See "SQL Anywhere Monitor" [*SQL Anywhere Server - Database Administration*].

- **Removing resources** You can remove a monitored resource from the Monitor without stopping it. Previously you had to stop the resource before you could remove it. See "Remove resources" [*SQL Anywhere Server - Database Administration*].

- **Resources no longer have states** Now resources only have statuses. See "Overview dashboard" [*SQL Anywhere Server - Database Administration*].

- **Changes to alert status** When a condition that triggered an alert no longer exists, the alert status changes to **Inactive**. An alert with the status **Inactive** indicates that the condition that triggered the alert is no longer present, but a Monitor user has not manually resolved it. In addition, the Monitor assigns a severity to each alert that it issues. See "Overview dashboard" [*SQL Anywhere Server - Database Administration*].

- **Changes to the alert threshold defaults** Previously, the Monitor issued alerts when the CPU memory usage reached 85 percent of the maximum cache size for two contiguous collection intervals. Now, the default is 90 percent and the time threshold is 30 seconds. See "Alert thresholds" [*SQL Anywhere Server - Database Administration*].

- **All users are required to log in into the Monitor** Previously, the Monitor did not require anyone to log in to have read-only access. Now all users are required to log in to the Monitor. See "Monitor users" [*SQL Anywhere Server - Database Administration*].

- **Changes to user language preferences** When a user logs in, the Monitor uses the language preferences of that user to configure the language displayed in the Monitor and used in alerts.

Previously, the Monitor used the language set in the browser. See "Edit Monitor users" [*SQL Anywhere Server - Database Administration*].

Documentation enhancements

Following is a list of changes made to the SQL Anywhere documentation in version 12.0.0.

- **Frequently Asked Questions (FAQ)** A comprehensive list of frequently asked questions is now available. Intended primarily for new and intermediate users, the FAQ attempts to answer many of the common questions that are asked regularly in newsgroup discussions. See "Frequently asked questions - SQL Anywhere" [*SQL Anywhere 12 - Introduction*].

- **Database server performance warnings** Descriptions are now provided for the performance warnings that appear in the database server messages window, as well as links to information about how to address the performance issue. See "Understanding database server performance warnings" [*SQL Anywhere Server - Database Administration*].

- **DocCommentXchange is the default documentation** DocCommentXchange is the default documentation format for SQL Anywhere 12. DocCommentXchange is an online community for accessing and discussing SQL Anywhere documentation on the web.

  If you prefer to install a local copy of the documentation, go to www.sybase.com/detail?id=1069195.

- **New Developer Centers on Sybase.com** To supplement the SQL Anywhere documentation, consider visiting the SQL Anywhere Developer Centers at http://www.sybase.com/developer/library/sql-anywhere-techcorner to browse technical white papers, FAQs, tech notes, downloads, techcasts and more for answers to your questions.

What's new in version 11.0.1

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

SQL Anywhere new features

Following is a list of additions to SQL Anywhere databases and database servers introduced in version 11.0.1.

Connection properties

The following connection properties have been added in SQL Anywhere version 11.0.1:

- Authenticated
- IsDebugger
- QueryBypassedCosted
- QueryBypassedHeuristic
- QueryBypassedOptimized
- QueryOpened
- QueryDescribedBypass
- QueryDescribedOptimizer
- StatementDescribes
- StatementPostAnnotates
- StatementPostAnnotatesSimple
- StatementPostAnnotatesSkipped
- WaitStartTime
- WaitType

For descriptions of these properties, see "Connection properties" [*SQL Anywhere Server - Database Administration*].

Database properties

The following database properties have been added in SQL Anywhere version 11.0.1:

- Authenticated
- Prepares
- QueryBypassedCosted
- QueryBypassedHeuristic
- QueryBypassedOptimized
- QueryOpened
- QueryDescribedBypass
- QueryDescribedOptimizer
- StatementDescribes
- StatementPostAnnotates
- StatementPostAnnotatesSimple
- StatementPostAnnotatesSkipped

For descriptions of these properties, see "Database properties" [*SQL Anywhere Server - Database Administration*].

Database server properties

Following is a list of enhancements made to database server properties in SQL Anywhere version 11.0.1.

- ServerEdition

For descriptions of these properties, see "Database server properties" [*SQL Anywhere Server - Database Administration*].

Database utilities

Following is a list of enhancements made to database utilities in SQL Anywhere version 11.0.1.

- **Service utility (dbsvc) enhancements** On Windows, you can now create services for the MobiLink Relay Server (rshost), relay server outbound enabler (RSOE), and Volume Shadow Copy Service (dbvss11) using the Service utility. See "Service utility (dbsvc) for Windows" [*SQL Anywhere Server - Database Administration*].

- **Unload utility (dbunload) enhancements** The dbunload utility now supports the following options:

  - **-cm option** Displays the dbinit command or CREATE DATABASE statement for the database being unloaded.

  - **-l option** Retains the next available value for autoincrement columns in the rebuilt database.

  See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

System procedures and functions

Following is a list of system procedure and function enhancements added in SQL Anywhere version 11.0.1.

- **sa_get_table_definition system procedure** The new sa_get_table_definition system procedure returns a LONG VARCHAR string containing the SQL statements required to create the specified table and its indexes, foreign keys, triggers and granted permissions. See "sa_get_table_definition system procedure" [*SQL Anywhere Server - SQL Reference*].

- **FIRST_VALUE function [Aggregate]** The FIRST_VALUE function [Aggregate] now includes a RESPECT NULLS clause. See "FIRST_VALUE function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].

- **LAST_VALUE function [Aggregate]** The LAST_VALUE function [Aggregate] now includes a RESPECT NULLS clause. See "LAST_VALUE function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].

- **sa_set_http_option system procedure** The AcceptCharset option now allows more control over character set selection. See "sa_set_http_option system procedure" [*SQL Anywhere Server - SQL Reference*].

SQL statements

Following is a list of SQL enhancements introduced in SQL Anywhere version 11.0.1.

- **New DEFAULT VALUES clause, INSERT statement** The new DEFAULT VALUES clause of the INSERT statement allows you to insert the default values for all columns. See "INSERT statement" [*SQL Anywhere Server - SQL Reference*].

- **CREATE ENCRYPTED DATABASE statement** This statement creates an encrypted copy of an existing database, including all transaction logs, mirror logs, and dbspaces. You can also use this statement to create a copy of a database with table encryption enabled. See "CREATE ENCRYPTED DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

  If you want to encrypt a database that requires recovery, for example to send to technical support, you must still use the CREATE ENCRYPTED FILE statement. See "CREATE ENCRYPTED FILE statement" [*SQL Anywhere Server - SQL Reference*].

- **CREATE DECRYPTED DATABASE statement** This statement creates a decrypted copy of an existing database, including all transaction logs, mirror logs, and dbspaces. See "CREATE DECRYPTED DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

  If you want to decrypt a database that requires recovery, for example to send to technical support, you must still use the CREATE DECRYPTED FILE statement. See "CREATE DECRYPTED FILE statement" [*SQL Anywhere Server - SQL Reference*].

- **ALTER DATABASE statement enhancement** Attempting to execute an ALTER DATABASE UPGRADE statement on a database server that is currently being mirrored now results in an error. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **MESSAGE statement enhancement** The IMMEDIATE clause causes the message to be received by the client's message callback routine within a short period of time, regardless of whether

the connection is idle or making requests. See "MESSAGE statement" [*SQL Anywhere Server - SQL Reference*].

- **Create or replace a function, procedure, trigger, or view of the same name** The new OR REPLACE clause allows you to create or replace a function, procedure, trigger, or view of the same name. See:
  - "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE TRIGGER statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE VIEW statement" [*SQL Anywhere Server - SQL Reference*]
- **Suppressing errors when a statement attempts to remove a database object that does not exist** The new IF EXISTS clause allows you to specify that you do not want an error returned when the DROP statement attempts to remove a database object that does not exist. See:
  - "DROP EVENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP FUNCTION statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP TABLE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP TRIGGER statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP VIEW statement" [*SQL Anywhere Server - SQL Reference*]
- **New INTO LOCAL TEMPORARY TABLE clause, SELECT statement** The new INTO LOCAL TEMPORARY TABLE clause of the SELECT statement allows you to create and populate a local temporary table with the result set of a SELECT statement. Previously, you could only do this by using an INTO clause if the temporary table name started with #. See "SELECT statement" [*SQL Anywhere Server - SQL Reference*].
- **New IF NOT EXISTS clause, CREATE TABLE statement** The new IF NOT EXISTS clause of the CREATE TABLE statement allows you to create permanent, global temporary, and local temporary tables if the table does not already exist. See "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*].
- **Specify an owner when creating temporary procedures or functions** The CREATE FUNCTION and CREATE PROCEDURE statements now allow you to optionally specify the owner for a temporary procedure or function. See:
  - "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*]

Programming interfaces

Following is a list of enhancements made to programming interfaces in SQL Anywhere version 11.0.1.

- **New ASP.NET providers** The following new ASP.NET providers mimic the functionality of the standard ASP.NET providers, but store their data in a SQL Anywhere database rather than in a SQL Server database:

- **Memberships** Allows logging in and out, managing users and their password
- **Roles** Allows assigning users to groups, allowing simple and easy permission management
- **Profiles** Stores user variables
- **Web Parts Personalization** Manages the storage of Web Parts data, allowing users to personalize their view
- **Web Events** Works with Health Monitoring to store flushed web event information in the database

See "SQL Anywhere ASP.NET Providers" [*SQL Anywhere Server - Programming*].

- **Support for Ruby** SQL Anywhere now supports the Ruby open source programming language. See "SQL Anywhere Ruby API support" [*SQL Anywhere Server - Programming*].
- **OLE DB now supports CATALOGS and SCHEMATA rowsets** The CATALOGS and SCHEMATA rowsets for OLE DB are now supported. Since SQL Anywhere does not support the notion of catalogs the SQL Anywhere OLE DB provider returns a result set for CATALOGS containing all currently started databases, and their locations, instead. Likewise, for SCHEMATA, the database name is used as the catalog in the result set.
- **Support added for ADO.NET Entity Framework** SQL Anywhere now provides support for the ADO.NET Entity Framework model. See "SQL Anywhere .NET Data Provider features" [*SQL Anywhere Server - Programming*].

Miscellaneous

Following is a list of miscellaneous enhancements introduced in SQL Anywhere version 11.0.1.

- **Improved performance for simple, inexpensive statement execution** The cost of generating an execution plan for statements with inexpensive execution times can sometimes be greater than the cost of executing the statement. With the following enhancements, SQL Anywhere now recognizes an expanded class of simple statements with inexpensive execution times and allows them to bypass the optimizer.

  For more information, see "Eligibility to skip query processing phases" [*SQL Anywhere Server - SQL Usage*].

  - **New FORCE NO OPTIMIZATION clause** Previously, all statements that required cost-based optimization were processed by the optimizer. With the addition of the FORCE NO OPTIMIZATION clause, you can specify that you want the statement to bypass the optimizer. If the statement is too complex to process in this way—possibly due to the setting of database options or characteristics of the schema or query—the statement fails and the database server returns an error.

The following statements support the new FORCE NO OPTIMIZATION clause:

- "DELETE statement" [*SQL Anywhere Server - SQL Reference*]
- "INSERT statement" [*SQL Anywhere Server - SQL Reference*]
- "SELECT statement" [*SQL Anywhere Server - SQL Reference*]
- "UPDATE statement" [*SQL Anywhere Server - SQL Reference*]

- **New connection properties**

  - QueryBypassedCosted
  - QueryBypassedHeuristic
  - QueryBypassedOptimized
  - QueryOpened
  - QueryDescribedBypass
  - QueryDescribedOptimizer
  - StatementDescribes
  - StatementPostAnnotates
  - StatementPostAnnotatesSimple
  - StatementPostAnnotatesSkipped

- **New database properties**

  - Prepares
  - QueryBypassedCosted
  - QueryBypassedHeuristic
  - QueryBypassedOptimized
  - QueryOpened
  - QueryDescribedBypass
  - QueryDescribedOptimizer
  - StatementDescribes
  - StatementPostAnnotates
  - StatementPostAnnotatesSimple
  - StatementPostAnnotatesSkipped

- **New Optimization Method field** The Optimization Method field has been added to the Optimizer Statistics section of the graphical plan. The field returns the execution strategy chosen by the query optimizer. See "Optimizer Statistics field descriptions" [*SQL Anywhere Server - SQL Usage*].

- **Prevent a database server from becoming the default database server** The -xd server option prevents a database server from listening on the default TCP port, and prevents the database server from becoming the default database server. See "-xd dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Support for parallel archive backups** The SQL Anywhere database server now supports parallel backups for server-side archive backups. Parallel database backups take advantage of physical I/O to perform read and write operations in parallel, instead of sequentially, which improves performance.

Two new clauses have been added to the BACKUP DATABASE statement to support parallel archive backups:

- **WITH CHECKPOINT LOG** [**NO**] **COPY**
- **MAX WRITE** { *n* | **AUTO** }

Version 11.0.0 and earlier database servers cannot restore archive backups generated with version 11.0.1 database servers. Version 11.0.1 database servers can restore backups produced by older database servers.

See "BACKUP statement" [*SQL Anywhere Server - SQL Reference*].

- **Running Developer Edition and Evaluation database servers on Mac OS X** You can now automatically start Developer Edition and Evaluation Edition database servers from the administration tools on Mac OS X.

SQL Anywhere behavior changes and deprecated features

Following is a list of changes to SQL Anywhere databases and database servers introduced in version 11.0.1, grouped by category.

Behavior changes

- **Full text searching** The following behavior changes have been made to full text searching:
  - **Operator precedence is now applied** Previously, no precedence was applied to operators in a query string. Now, the following operator precedence is applied:
    - NEAR, FUZZY operators
    - AND NOT operator
    - AND operator
    - OR operator

    See "Operator precedence in a CONTAINS search condition" [*SQL Anywhere Server - SQL Reference*].
  - **NEAR clause arguments must be terms or prefix terms** When you perform a proximity search, the arguments to the NEAR clause must be terms or prefix terms. See "CONTAINS search condition" [*SQL Anywhere Server - SQL Reference*], and "Proximity searching" [*SQL Anywhere Server - SQL Usage*].
  - **Use of hyphen and AND NOT clause** Within a phrase, a hyphen is treated as a term breaker, not a special character. Outside of a phrase, the treatment of a hyphen depends on the syntax surrounding the hyphen. See "Allowed syntax for hyphen (-)" [*SQL Anywhere Server - SQL Reference*], and "Using the AND NOT operator in full text searches" [*SQL Anywhere Server - SQL Usage*].

- **Use of asterisk and prefix searching** When performing a prefix search, the asterisk must be appended to a term and followed immediately by a space, or by the end of query string, or by one of the allowed special characters. See "Allowed syntax for asterisk (*)" [*SQL Anywhere Server - SQL Reference*], and "Prefix searching" [*SQL Anywhere Server - SQL Usage*].

- **Creating a duplicate text index now returns an error** You can no longer create duplicate text indexes. A text index is considered a duplicate if the following settings are identical to those of an existing text index:

  - the base table being referenced
  - the columns to be indexed (order does not matter)
  - the settings for the configuration object used (TERM BREAKER, MINIMUM TERM LENGTH, MAXIMUM TERM LENGTH, STOPLIST, collation information)

  Duplicate text indexes created before SQL Anywhere 11.0.1 can remain in the database and do not cause errors when started with a 11.0.1 database server. However, if a database that contains duplicate text indexes is reloaded on version 11.0.1 or later, an error is returned.

  To identify duplicate text indexes in an existing database, execute the following query:

```
SELECT LIST( i.index_name )
FROM SYS.SYSIDX i
    JOIN SYS.SYSTEXTIDX t ON i.object_id = t.index_id AND t.sequence =
1
    JOIN SYS.SYSTEXTCONFIG F ON F.object_id = t.text_config
    JOIN (
    SELECT table_id, index_id, LIST( column_id, ', ' ORDER BY
column_id ) col_id
    FROM SYS.SYSIDXCOL
    GROUP BY table_id, index_id) x
    ON x.table_id = i.table_id AND x.index_id = i.index_id
WHERE i.index_category=4
GROUP BY i.table_id, f.term_breaker, f.min_term_length,
f.max_term_length,
    f.collation, ISNULL( f.char_stoplist, '-' ),
    ISNULL( f.nchar_stoplist, '-' ), x.col_id
HAVING count(*) > 1
```

  This query works only if the strings representing STOPLIST are exactly the same or if NO STOPLIST is specified. For example, stoplists 'a b c' and 'a - b c' are not considered the same stoplist by this query, but would be considered the same during a check for duplicates during text index creation.

- **Regular expressions** Changes have been made to the behavior for the SIMILAR TO and REGEXP search conditions, and the REGEXP_SUBSTR function. The overall intention of the changes is to continue to have SIMILAR TO be consistent with the ANSI/SQL standard, while making the behavior of REGEXP and REGEXP_SUBSTR consistent with Perl.

  - **Database collations and matching** Previously, REGEXP and REGEXP_SUBSTR determined if a literal or character class range in the pattern matched the string by using the collation equivalence and sort order. Now, REGEXP and REGEXP_SUBSTR use binary comparisons of code point values for matching and evaluation of ranges. The change was made to make the behavior consistent with Perl 5.0.

SIMILAR TO still uses the database collation for matching and range evaluation. See "LIKE, REGEXP, and SIMILAR TO search conditions" [*SQL Anywhere Server - SQL Reference*].

- **Database case sensitivity and [[:upper:]] and [[:lower:]] sub-character classes** SIMILAR TO and REGEXP [[:upper:]] and [[:lower:]] sub-character classes were case-insensitive on a case-insensitive database. This has been changed so that [[:upper:]] only matches upper case characters and [[:lower:]] only matches lower case characters, regardless of the database case sensitivity.

- **Treatment of caret (^), underscore (_), and percent sign (%) as metacharacters** The following table explains the previous and new treatment of these characters as metacharacters:

Character	Previous behavior	New behavior
_ (underscore)	For SIMILAR TO, REGEXP, and REGEXP_SUBSTR, an underscore was treated as a metacharacter: it matched any single character.	For SIMILAR TO, an underscore is treated as a metacharacter: it matches any single character. For REGEXP and REGEXP_SUBSTR, an underscore is not treated as a metacharacter. Instead, REGEXP, and REGEXP_SUBSTR use a period (.) to match any single character.
%	For SIMILAR TO, REGEXP, and REGEXP_SUBSTR, a percent sign was treated as a metacharacter: it matched any number of any characters.	For SIMILAR TO, a percent sign is treated as a metacharacter: it matches any number of any characters. For REGEXP and REGEXP_SUBSTR, a percent sign is not treated as a metacharacter. Instead, REGEXP, and REGEXP_SUBSTR use dot-asterisk (.*) to match any number of any characters.

Character	Previous behavior	New behavior
^	For SIMILAR TO, REGEXP, and REGEXP_SUBSTR, a caret inside a character class was treated as a negation or subtraction character for anything to the right of it: it was interpreted as NOT matches.	For SIMILAR TO, a caret is treated as a negation or subtraction character for characters to the right of it. For example, `SIMILAR TO [a-d^c]` matches a, b, d, but not c. For REGEXP and REGEXP_SUBSTR, the caret is only treated as a metacharacter if it is in the first position inside a character class: it is interpreted as a negation of the character class. For example, `REGEXP [^abc]` matches any single character that is not a, b, or c, while `REGEXP [a-d^c]` matches a, b, c, d, and ^.

- **Upgrade utility (dbupgrad) behavior change** Attempting to use the Upgrade utility to upgrade a database that is participating in database mirroring now results in an error. See "Upgrade utility (dbupgrad)" [*SQL Anywhere Server - Database Administration*].

- **Mac OS X no longer requires dbmodenv** In previous releases, to use the graphical administration tools on Mac OS X, the user's *$HOME/.MacOSX/environment.plist* file had to have the SQL Anywhere binary and library location added to PATH and DYLD_LIBRARY_PATH. You could do this using the dbmodenv tool. SQL Anywhere no longer depends on the settings in *$HOME/.MacOSX/environment.plist*, and you no longer need to run dbmodenv or log out and log in again after installing SQL Anywhere.

- **Default changed for null values returned from the dbisqlc OUTPUT statement** In previous releases, if you used the OUTPUT statement from dbisqlc, the statement returned the value **(NULL)** for null values. Now, the statement returns an empty string by default for null values. You can change the way NULL values are exported by setting the output_nulls option. See "output_nulls option [Interactive SQL]" [*SQL Anywhere Server - Database Administration*].

- **Endian support** After upgrading, pre-11.0.1 text indexes created on a big-endian computer need to be truncated and refreshed (for MANUAL REFRESH and AUTO REFRESH text indexes) or recreated (for IMMEDIATE REFRESH indexes).

- **Administration tools on Mac OS X** On Mac OS X, the SQL Anywhere administration tools now use the 64-bit JDK 1.6. The administration tools only run on Intel-based Macintosh computers with 64-bit processors supported by the Apple JDK 1.6 (Mac OS X 10.5.2 or later). If you are deploying the

administration tools for Mac OS X, the native libraries are located in *$SQLANY11/System/lib64*. See "Deploying administration tools on Linux, Solaris, and Mac OS X" [*SQL Anywhere Server - Programming*].

- **New default size for chunked transfer-coding for HTTP clients** Previously, if an HTTP client sent data greater than 2048 bytes, chunked transfer-coding was attempted by default (or if the user specifies `CREATE PROCEDURE ... SET 'HTTP(CH=auto)'`. The default size has been changed from 2048 to 8196 bytes. Also, a new status, 411 **Length Required**, has been added to the criteria for re-issuing the request without using chunked transfer-coding. See "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*].

- **ansi_substring option support** The ansi_substring option was deprecated in version 11.0.0, but is now supported for version 11.0.1. See "ansi_substring option" [*SQL Anywhere Server - Database Administration*].

Deprecated and discontinued features

- **COMMENT ON EXTERNAL ENVIRONMENT OBJECT *object-name*** The syntax has been changed to **COMMENT ON EXTERNAL OBJECT** *object-name*. Currently, the old syntax is still accepted but may not be supported in a future release. See "COMMENT statement" [*SQL Anywhere Server - SQL Reference*].

MobiLink new features

Following is a list of additions to MobiLink introduced in version 11.0.1.

- **Schema caching of remote database schema** The new schema caching feature reduces overhead for smaller synchronizations. Remote schemas are cached by the MobiLink server on the first synchronization. On subsequent synchronizations, remote schema information is only sent to the MobiLink server if it does not already have the schema cached.

- **-vi option for mlsrv11** Display the column values of each row uploaded. See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **-vq option for mlsrv11** Display the column values of each row downloaded. See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **-vm option for mlsrv11** Prints the duration of each synchronization and the duration of each synchronization phase to the log whenever a synchronization completes. See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **-ppv option for mlsrv11** Causes MobiLink to print new periodic monitoring values according to the period specified. See "-ppv mlsrv12 option" [*MobiLink - Server Administration*].

- **ml_ignore prefix** The MobiLink server recognizes SQL scripts prefixed with --{ml_ignore} as intentionally ignored scripts. See "Ignoring scripts" [*MobiLink - Server Administration*].

- **-sv option for dblsn** Specifies the script version used by the MobiLink listener to authenticate against a database. See "-sv dblsn option" [*MobiLink - Server-Initiated Synchronization*].
- **Support for Oracle VArray** The iAnywhere Solutions 11 - Oracle ODBC driver now supports the use of Oracle VArray in stored procedures. See "Using Oracle VARRAY" [*MobiLink - Server Administration*].
- **Light weight polling listener keywords variables** The poll_connect, poll_notifier, poll_key, and poll_every listener keywords have been added to support light weight polling.
- **Light weight polling action variables** $poll_connect, $poll_notifier, $poll_key, and $poll_every action variables have been added to support light weight polling.
- **Client authentication using Common Access Cards** MobiLink clients now support authentication using client identities from Common Access Cards (CACs). See "identity_name" [*MobiLink - Client Administration*].

  Separately licensed component required
  This feature is part of the CAC Authentication Add-on and requires a separate license. See "Separately licensed components" [*SQL Anywhere 12 - Introduction*].
- **Support for Microsoft SQL Server 2008** The MobiLink synchronization server now supports consolidated databases running on Microsoft SQL Server 2008. For information about mapping the new Microsoft DATE, TIME, and DATETIME2 data types, see "Microsoft SQL Server data mapping" [*MobiLink - Server Administration*].
- **New method for .NET DownloadTableData interface** getLastDownloadTime method to return last download time for a table. See "GetLastDownloadTime method" [*MobiLink - Server Administration*].
- **Log verbosity for targeted MobiLink users and remote IDs** You can now set different log verbosity for a targeted MobiLink user or remote ID. See "Log verbosity for targeted MobiLink users and remote IDs" [*MobiLink - Server Administration*].
- **Support for MySQL in Model mode** The MobiLink plug-in now supports MySQL consolidated databases.
- **-c option for mlmon** The -c option has been added for the mlmon command for the MobiLink Monitor. The -c option closes the MobiLink Monitor at the end of the connection and saves the session to the specified database.

MobiLink behavior changes and deprecated features

Following is a list of changes to MobiLink introduced in version 11.0.1.

- **-sm option for mlsrv11** The -sm option for mlsrv11 has been improved to provide similar functionality to the -nc option, when used with non-persistent HTTP/HTTPS. See "-sm mlsrv12

option" [*MobiLink - Server Administration*] and "-nc mlsrv12 option" [*MobiLink - Server Administration*].

- **Microsoft SQL Server data types** As of SQL Server 2005, TEXT, NTEXT, and IMAGE types are deprecated. Use VARCHAR(*max*), NVARCHAR(*max*) and VARBINARY(*max*) instead. See "Microsoft SQL Server data mapping" [*MobiLink - Server Administration*].

QAnywhere new features

Following is a list of additions to QAnywhere introduced in version 11.0.1.

- **QAnywhere standalone client** The QAnywhere standalone client provides a compact client that enables you to set up a messaging system without worrying about running the QAnywhere Agent or administering your database. See "QAnywhere standalone client" [*QAnywhere*].

UltraLite new features

Following is a list of additions to UltraLite introduced in version 11.0.1.

- **Mirror file** UltraLite provides basic database file mirroring to improve fault tolerance on potentially unreliable storage systems. This is accomplished using the mirror file. All database writes are issued to the mirror file at the same time as they are to the main database file. See "UltraLite MIRROR_FILE connection parameter" [*UltraLite - Database Management and Reference*].

- **Unset ml_remote_id** You can now unset **ml_remote_id** using SET OPTION. See "SET OPTION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*] and "ML_GET_SERVER_NOTIFICATION [System]" [*UltraLite - Database Management and Reference*].

- **ML_GET_SERVER_NOTIFICATION** This function allows UltraLite users to use light weight polling to query a notifier on a MobiLink server for server-initiated synchronization requests. See "ML_GET_SERVER_NOTIFICATION [System]" [*UltraLite - Database Management and Reference*].

- **SYNC_PROFILE_OPTION_VALUE function** This function returns the value of specified options within a sync profile. See "SYNC_PROFILE_OPTION_VALUE function [System]" [*UltraLite - Database Management and Reference*].

- **StreamErrorParameters property for the UltraLite.NET API** **StreamErrorParameters** has been added to the **SyncResult** class. This member contains a comma separated list of error parameters for the stream error code reported in StreamErrorCode. See "StreamErrorParameters property" [*UltraLite - .NET Programming*].

- **getStreamErrorParameters method for the UltraLite for M-Business API** This method returns a comma separated list of error parameters reported by the synchronization stream processing. See "getStreamErrorParameters method" [*UltraLite - M-Business Anywhere Programming*].

UltraLite behavior changes and deprecated features

Following is a list of changes to UltraLite introduced in version 11.0.1.

- **Palm Suspend functionality** Has been deprecated.

UltraLiteJ new features

Following is a list of additions to UltraLiteJ introduced in version 11.0.1.

- **Additional SQL support** UltraLiteJ provides additional SQL support as follows:
  - ALTER TABLE
  - CREATE TABLE
  - CREATE INDEX
  - DROP INDEX
  - DROP TABLE
  - TRUNCATE TABLE

  In addition, the following restrictions have been removed:
  - HAVING is supported.
  - DISTINCT within aggregate functions is supported.
  - CURRENT TIME, CURRENT TIMESTAMP and CURRENT DATE are supported.

  See "Supported SQL statements" [*UltraLiteJ*].
- **New DatabaseInfo methods** Two new methods, **getPageReads()** and **getPageWrites**, have been added to the **DatabaseInfo** interface. These methods return the number of page reads and writes at the time a DatabaseInfo object was created. See "getPageReads method" [*UltraLiteJ*] and "getPageWrites method" [*UltraLiteJ*].
- **Updated UltraLiteJ Database Transfer utility** The UltraLiteJ Database Transfer utility now provides the ability to delete a database, display database information, or view or email the database

transfer log, as well as transfer a database from the client. See "UltraLiteJ Database Transfer utility (ULjDbT)" [*UltraLiteJ*].

UltraLiteJ behavior changes and deprecated features

Following is a list of changes to UltraLiteJ introduced in version 11.0.1.

- **@@identity global variable** The **@@identity** global variable is not supported by UltraLiteJ.

Administration tool new features

Following is a list of additions to Sybase Central and Interactive SQL introduced in version 11.0.1.

Sybase Central new features

Following is a list of additions to Sybase Central plug-ins introduced in version 11.0.1.

All plug-ins

- **Create Service Wizard enhancements** You can now create services for the MobiLink Relay Server (rshost), relay server outbound enabler (RSOE), Volume Shadow Copy Service (dbvss11), MobiLink Listener utility (dblsn), and the Broadcast Repeater utility (dbns11) using the **Create Service Wizard**. See "Programs that can be run as Windows services" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere plug-in

- **View the status of events in Sybase Central** Now you can see the current running state of events in Sybase Central. In the **Events** folder there is a new **Running** column that shows the current running state of an event. The contents of the folder are updated whenever the folder is selected in the left-pane. In addition there is a **Running** property in the **Event Properties** window.

  The running value shows **Yes** if the event is running, **No** if the event is not running, or **Unknown** if the event exists in a SQL Anywhere 9.0.2 or earlier database.

- **View the status of maintenance plans in Sybase Central** Now you see the current running state of maintenance plans in Sybase Central. In the **Maintenance Plans** folder there is a new **Running** column that shows the current running state of a plan. The contents of the folder are updated whenever the folder is selected in the left-pane.

  The running value shows **Yes** if the maintenance plan is running, **No** if the plan is not running, or **Unknown** if the plan exists in a SQL Anywhere 9.0.2 or earlier database.

- **Maintenance plan enhancements** The **Create Maintenance Plan Wizard** now lets you send a test email when you configure the settings to email a maintenance plan report. See "Creating a maintenance plan" [*SQL Anywhere Server - Database Administration*].
- **Create Procedure Wizard and Create Function Wizard enhancements** Now in the **Create Function Wizard** and the **Create Procedure Wizard**, you can choose one of the following SQL dialects or languages to write the procedure or function in: Watcom-SQL, Transact-SQL, External C/C++, or External Environment. If you choose External Environment, you select one of the following languages: C_ESQL32, C_ESQL64,C_ODBC32, C_ODBC64, CLR, JAVA, PERL, or PHP. Previously, only Watcom-SQL or Transact-SQL were available choices. Choosing C/C++ or Java generates a code skeleton for the function or procedure with the EXTERNAL NAME clause. See "CREATE FUNCTION statement (external procedures)" [*SQL Anywhere Server - SQL Reference*] and "CREATE PROCEDURE statement (external procedures)" [*SQL Anywhere Server - SQL Reference*]

MobiLink plug-in

- **MobiLink model enhancement for Oracle databases** Now, when you use the **Create Synchronization Model Wizard** to create a synchronization model that uses an Oracle consolidated database, you can choose to load the entire schema, or to select a subset of owners whose tables you need for synchronization. Choosing a subset of owners can reduce the schema loading time.
- **MobiLink Model mode support for MySQL** MobiLink Model mode now supports the use of MySQL consolidated databases.

Administration tool behavior changes and deprecated features

Following is a list of changes to Sybase Central and Interactive SQL introduced in version 11.0.1.

Sybase Central behavior changes and deprecated features

Following is a list of changes to Sybase Central introduced in version 11.0.1.

SQL Anywhere plug-in

- **Read-only database enhancement** Now you receive a warning when you connect to a read-only database. You can choose to suppress this warning by choosing **Tools** » **SQL Anywhere 11** » **Preferences**.

Interactive SQL behavior changes and deprecated features

Following is a list of changes to Interactive SQL introduced in version 11.0.1.

- **Configure the automatic release of database locks** Now, you can configure Interactive SQL to attempt to release the database schema locks it creates when it displays your result set. To do so, in

Interactive SQL, choose **Tools** » **Options** » **SQL Anywhere**, and select **Automatically Release Database Locks**.

When this option is selected, after you execute a statement that returns a result set, Interactive SQL checks if your connection has any uncommitted changes in the database. If none exist, then Interactive SQL releases your schema locks; otherwise, Interactive SQL does not release your schema locks. That is, Interactive SQL does not release your schema locks if you have any uncommitted changes to the database.

Product-wide new features

Following is a list of product-wide additions introduced in version 11.0.1.

- **New online documentation forum, DocCommentXchange (DCX)** A new online forum called DocCommentXchange has been created. DocCommentXchange is a community for accessing and discussing SQL Anywhere documentation.

  Use DocCommentXchange to:

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

  Visit http://dcx.sybase.com.

- **SQL Anywhere Monitor** The SQL Anywhere Monitor is a browser-based administration tool that provides you with information about the health and availability of SQL Anywhere databases and MobiLink servers.

  For MobiLink users, this feature does not replace or overlap the functionality in the existing MobiLink Monitor.

  See “SQL Anywhere Monitor” [*SQL Anywhere Server - Database Administration*].

Documentation enhancements

- **SQL Anywhere backup and recovery documentation enhancements** The documentation for using the backup and recovery tools included with SQL Anywhere has been re-written for this release. See “Backup and data recovery” [*SQL Anywhere Server - Database Administration*].

- **Full text search documentation enhancements** The documentation for the full text feature has been reorganized and now contains more examples and tutorials. See “Full text search” [*SQL Anywhere Server - SQL Usage*].

- **MobiLink Server-Initiated Synchronization book** The MobiLink server-initiated synchronization book has undergone improvements since version 11.0.0. For example, the table of contents organization has been improved, server-initiated synchronization is explained in more detail, and improvements have been made to the format of the reference material. See "MobiLink - Server-Initiated Synchronization".

- **SQL Remote book** The SQL Remote book has been reorganized and re-written to enhance navigation and usability. See "SQL Remote".

What's new in version 11.0.0

For information about new features and behavior changes in versions of SQL Anywhere before version 9, go to http://dcx.sybase.com/html/dbwnen10/dbwnen10.html.

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

SQL Anywhere

MobiLink

QAnywhere

SQL Remote

UltraLite

Sybase Central and Interactive SQL

Documentation enhancements

Product-wide features

SQL Anywhere

The following sections describe the new features, behavior changes, and deprecated features in SQL Anywhere for version 11.0.0.

SQL Anywhere new features

Following is a list of additions to SQL Anywhere databases and database servers introduced in version 11.0.0.

Main features

Following is a list of the main features introduced in SQL Anywhere version 11.0.0.

- **Support for merging tables** SQL Anywhere now allows you to merge tables, views, and system procedure results into a table or view. See "MERGE statement" [*SQL Anywhere Server - SQL Reference*].

- **Support for login policies** SQL Anywhere now supports login policies. A login policy is a set of options that define rules to be applied when a database connection is established for a user. SQL Anywhere supplies a root policy used to store default values for all login policies. Separate login policies can be created to store overrides to the root login policy options. Each user is assigned a login policy and does not inherit any of the login policies through group memberships. The assignment of a policy to a specific user can be modified as necessary. See "Managing login policies" [*SQL Anywhere Server - Database Administration*].

  A database upgrade is required to take advantage of this feature. See "Upgrading SQL Anywhere" on page 304.

- **Support for full text search** SQL Anywhere now supports full text search. Full text search can quickly find all instances of a term (word) in a database. Full text search differs from searching using predicates such as LIKE, REGEXP, and SIMILAR TO because it is term-based and because it uses a text index instead of scanning table rows. See "Full text search" [*SQL Anywhere Server - SQL Usage*].

  You must upgrade your database to make use of the full text search feature. See "Upgrading SQL Anywhere" on page 304.

- **Support for regular expressions** SQL Anywhere now provides support for regular expressions using two new search conditions, REGEXP and SIMILAR TO. See "REGEXP search condition" [*SQL Anywhere Server - SQL Reference*] and "SIMILAR TO search condition" [*SQL Anywhere Server - SQL Reference*].

  See also, "Regular expressions overview" [*SQL Anywhere Server - SQL Reference*] and "LIKE, REGEXP, and SIMILAR TO search conditions" [*SQL Anywhere Server - SQL Reference*].

- **Database option settings are now recorded in the transaction log** The database option settings in effect during a LOAD operation are now recorded in the transaction log. This ensures that there are no inconsistencies in the data should options, such as date_order and nearest_century, change

between the original LOAD operation and the final LOAD operation resulting from applying the transaction log during recovery.

- **Enhancements to the optimizer's use of indexes** Several enhancements have been made to the indexing capabilities of SQL Anywhere. You must upgrade your database to make use of these new features. See "Upgrading SQL Anywhere" on page 304.

  - **Support for multiple indexes scan** The optimizer has been enhanced to consider multiple indexes (up to four) to retrieve data from a base table based on multiple predicates on that table. Previously, you could only specify one index as an index hint for a query. A new index hint in the WITH clause of the SELECT statement allows you to specify that a multiple index scan can be used. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

    A new table access algorithm, Multiple Index Scan, has been added. See "MultipleIndexScan method (MultIdx)" [*SQL Anywhere Server - SQL Usage*].

  - **Support for index-only retrieval** The optimizer has been enhanced to support index-only retrievals. With index-only retrieval, the query is satisfied using data from the indexes, without having to access corresponding rows in the tables. The optimizer always performs an index-only retrieval when possible. An INDEX ONLY { ON | OFF } hint can be used to control whether index-only retrieval is performed. See "Indexes can be used to satisfy a predicate" [*SQL Anywhere Server - SQL Usage*] and "FROM clause" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to loading and unloading data** The following enhancements have been made to loading and unloading data:

  - **Load data from, and unload data to, files on a client computer** SQL statements and functions are used to read and write data residing on the database server. New features have been implemented to extend this capability to files that reside on the client computer, without the need to copy client files onto the database server. The transfer of data is accomplished efficiently, while providing security and access control for data on the client computer.

    The actual reading of files on the client computer is done transparently by the client libraries, which means that existing client applications can start benefitting immediately from the new feature using the new SQL language support.

    To benefit from these new capabilities, both the client and the database server must be SQL Anywhere version 11.0.0, and the client must use the Command Sequence communication protocol (CmdSeq).

    See "Accessing data on client computers" [*SQL Anywhere Server - SQL Usage*].

  - **Unload data into a variable** The UNLOAD statement has been enhanced to include an INTO VARIABLE clause to allow you to unload data into a variable. See "UNLOAD statement" [*SQL Anywhere Server - SQL Reference*].

  - **Load data from a column in another table** The LOAD TABLE statement has been enhanced to include a USING COLUMN clause to allow you to load data from a column in another table. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*] and "Import data with the LOAD TABLE statement" [*SQL Anywhere Server - SQL Usage*].

A database upgrade is required to take advantage of this new feature. See "Upgrading SQL Anywhere" on page 304.

- **Load data from a value (BLOB)** The LOAD TABLE statement has been enhanced to include a USING VALUE clause to allow you to load data from a value expression, such as the results of a function or a system procedure. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*], and "Import data with the LOAD TABLE statement" [*SQL Anywhere Server - SQL Usage*].

  A database upgrade is required to take advantage of this new feature. See "Upgrading SQL Anywhere" on page 304.

- **LOAD TABLE statement recovery and mirroring enhancements** Previously, in a mirrored database configuration, loading data from a file using the LOAD TABLE statement was not supported because only the LOAD TABLE statement was recorded in the transaction log, not the data being loaded. Additionally, when recovering a database, data loaded using a LOAD TABLE statement was not recoverable unless the original load file was present during recovery.

  The LOAD TABLE statement has been enhanced to include three new logging option clauses: WITH CONTENT LOGGING, WITH ROW LOGGING, and WITH FILE NAME LOGGING. These clauses allow you to control whether to record the loaded data in the transaction log. In a database mirroring system, that data can be used to load the mirror database. Additionally, during recovery, the load file no longer needs to be present. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

  A database upgrade is required to take advantage of this feature. See "Upgrading SQL Anywhere" on page 304.

- **Enhancements to materialized views** Support for materialized views has been enhanced as follows:

  - **Support for immediate materialized views** You can now configure materialized views to be refreshed immediately when data changes in the underlying tables impact data in the materialized view. Views with this refresh type are called **immediate views**; views that are not refreshed immediately are now referred to as **manual views**. Materialized views created before this release are considered manual views, and are the default when creating a new materialized view.

    For more information about manual and immediate views, see "Working with materialized views" [*SQL Anywhere Server - SQL Usage*] and "Manual and immediate materialized views" [*SQL Anywhere Server - SQL Usage*].

    You must upgrade your database to use the system procedures that support this feature. See "Upgrading SQL Anywhere" on page 304.

  - **Ability to refresh multiple materialized views at once** Previously, you needed to refresh materialized views one at a time. Changes to underlying data between each refresh operation could introduce inconsistencies between the materialized views. Now, to refresh multiple materialized views using the same data, you can specify a list of materialized views for the REFRESH MATERIALIZED VIEW statement. See "REFRESH MATERIALIZED VIEW statement" [*SQL*

Anywhere Server - SQL Reference] and "Refresh manual views" [*SQL Anywhere Server - SQL Usage*].

- **New WITH SHARE MODE clause for the REFRESH MATERIALIZED VIEW statement** A new clause, WITH SHARE MODE, has been added to the REFRESH MATERIALIZED VIEW statement. This mode gives read access on underlying tables to other transactions while the refresh operation takes place. When this clause is specified, shared table locks are obtained on all underlying base tables before the refresh operation is performed. The default mode is now WITH SHARE MODE, unless the materialized view is defined as IMMEDIATE REFRESH, or snapshot isolation is enabled for the database. For more information about the default refresh behavior, see "REFRESH MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*].

- **Support for querying the contents of a file or BLOB string** Using the new OPENSTRING subclause of the FROM clause, you can now query data from a file or a BLOB string. The OPENSTRING clause allows you to specify the object to be queried, as well as the schema and other parsing information for the data. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

  A new plan item, OpenString, appears in the execution plan when the OPENSTRING operation is performed. See "OpenString algorithm (OpenString)" [*SQL Anywhere Server - SQL Usage*].

- **Improved support for compressed indexes** Because of the work to improve support for compressed indexes, when you rebuild a database by unloading and reloading it, the rebuilt database may be smaller than the original database. This decrease in database size does *not* indicate a problem or a loss of data.

- **Read-only access to databases running on a mirror server** If you are using database mirroring, you can now connect to the database running on the mirror server. This enables you to offload potentially resource-heavy reporting operations to the mirror server, while leaving the primary server available. You can connect to a mirror database by providing a database server name with the -sm server option that can be used to access the read-only mirror database. See "-sm dbsrv12 database option (deprecated)" [*SQL Anywhere Server - Database Administration*] and "Configuring read-only access to a database running on the mirror server" [*SQL Anywhere Server - Database Administration*].

Database connections

Following is a list of enhancements made to database connections in SQL Anywhere version 11.0.0.

- **AppInfo connection parameter enhancements** The AppInfo connection parameter now supports the OSUSER key. This key returns the operating system user name associated with the client process. Linux and Solaris now support the EXE key. See "AppInfo (APP) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Elevate connection parameter** The Elevate connection parameter elevates automatically started SQL Anywhere database servers on Windows Vista. See "Elevate connection parameter" [*SQL Anywhere Server - Database Administration*].

- **NewPassword connection parameter [NEWPWD]** The NewPassword connection parameter allows users to change passwords, even if they have expired, without DBA intervention. See

"NewPassword (NEWPWD) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Prefetch enhancements** The default values for the PrefetchBuffer (PBUF) connection parameter have changed. On Windows Mobile, the default value is now 64 KB, and on all other platforms the default value is now 512 KB. This connection parameter accepts values between 64 KB and 8 MB. See "PrefetchBuffer (PBUF) connection parameter" [*SQL Anywhere Server - Database Administration*].

  In previous versions, the maximum number of prefetched rows was based on the maximum amount of data that could be prefetched. Now, the maximum number of prefetched rows takes into account the actual amount of prefetched data, as well as the data limit specified by the PrefetchBuffer connection parameter. This can result in significant performance gains when the amount of data in a column is significantly less than both the host variable length and describe length.

  Prefetch also dynamically increases the number of prefetch rows in situations that are likely to result in improved performance. See "Prefetching rows" [*SQL Anywhere Server - Programming*].

Backup and recovery

Following is a list of backup and recovery enhancements introduced in SQL Anywhere version 11.0.0.

- **Support for the Microsoft Volume Shadow Copy Service (VSS)** SQL Anywhere is compatible with Microsoft Volume Shadow Copy Service (VSS). To use VSS, you must rebuild all existing databases. See "Using the SQL Anywhere Volume Shadow Copy Service (VSS)" [*SQL Anywhere Server - Database Administration*].

Security

Following is a list of security enhancements introduced in SQL Anywhere version 11.0.0.

- **ISYSUSER and ISYSEXTERNLOGIN system tables are now encrypted when table encryption is enabled** Previously, when encrypting a database, or when creating a database with table encryption enabled, the ISYSCOLSTAT system table was automatically encrypted. Now, the ISYSUSER and ISYSEXTERNLOGIN system tables are also encrypted, to provide additional security.

- **Auditing enhancements** Now, auditing can be controlled through Sybase Central. From the **Database Properties** window, users with DBA authority can enable auditing, disable auditing, and specify which information they want to audit. Auditing information can be viewed in Sybase Central on the **Auditing** tab in the right pane. See "Controlling auditing" [*SQL Anywhere Server - Database Administration*] and "Retrieving auditing information" [*SQL Anywhere Server - Database Administration*].

  When auditing is enabled, errors for failed connections are now logged, indicating the reason for the failure.

- **256-bit AES encryption now supported** SQL Anywhere now supports 256-bit AES encryption for databases, tables, files, and data. This enhancement impacts several areas, as noted below:

- **Database and table encryption** You can now specify AES256 and AES256_FIPS for the ENCRYPTION clause of the CREATE DATABASE statement. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

  You can also specify AES256 and AES256_FIPS for the -ea option of the Initialization utility (dbinit) and Unload utility (dbunload). See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*] and "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

- **FIPS-approved algorithms** You can now use a 256-bit FIPS-approved AES algorithm on a FIPS-enabled platform. See "-fips dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Encrypting and decrypting data** When encrypting data using the ENCRYPT and DECRYPT functions, you can now specify AES256 and AES256_FIPS. See "ENCRYPT function [String]" [*SQL Anywhere Server - SQL Reference*] and "DECRYPT function [String]" [*SQL Anywhere Server - SQL Reference*].

- **Creating encrypted copies of databases, transaction logs, and dbspaces** When creating an encrypted copy of an encrypted or unencrypted database, transaction log, or dbspace using the CREATE ENCRYPTED FILE statement, you can now specify a 256-bit AES algorithm (AES256 or AES256_FIPS). See "CREATE ENCRYPTED FILE statement" [*SQL Anywhere Server - SQL Reference*].

- **DBTools support for 256-bit AES encryption** The a_create_db and an_unload_db structures have been extended to support AES256 and AES256_FIPS as values for the encryption_algorithm member. See "a_create_db structure" [*SQL Anywhere Server - Programming*] and "an_unload_db structure" [*SQL Anywhere Server - Programming*].

  See also:

  - "Encrypting and decrypting a database" [*SQL Anywhere Server - Database Administration*]
  - "Database properties" [*SQL Anywhere Server - Database Administration*]

- **Password encryption supported for jConnect and Open Client** Password encryption is now supported for jConnect and Open Client connections. See:

  - "Using the jConnect JDBC driver" [*SQL Anywhere Server - Programming*]
  - "Deploying JDBC clients" [*SQL Anywhere Server - Programming*]
  - "Known Open Client limitations of SQL Anywhere" [*SQL Anywhere Server - Programming*]

Database permissions and authorities

The following list contains the new or enhanced permissions and authorities in SQL Anywhere. You must upgrade your database to make use of these changes. See "Upgrading SQL Anywhere" on page 304.

- **Support for inheriting some authorities** SQL Anywhere now supports the inheriting of some authorities, as follows:

- PROFILE, READCLIENTFILE, READFILE, and WRITECLIENTFILE are inheritable
- DBA, BACKUP, and RESOURCE authority are not inheritable

See "Authorities overview" [*SQL Anywhere Server - Database Administration*].

- **New PROFILE authority** Previously, a user needed DBA authority to perform application profiling and diagnostics tracing. Now, users with PROFILE authority can also perform these tasks. For the full list of permissions available to PROFILE authority, see "PROFILE authority" [*SQL Anywhere Server - Database Administration*].
- **New READFILE authority** The READFILE authority allows a user to select from a file using the OPENSTRING clause of a SELECT statement. See "READFILE authority" [*SQL Anywhere Server - Database Administration*] and "FROM clause" [*SQL Anywhere Server - SQL Reference*].
- **New READCLIENTFILE authority** The READCLIENTFILE authority allows a user read from a file on a client computer. For example, to use the LOAD TABLE statement to load data from a file on the client computer, the user needs READCLIENTFILE authority. See "READCLIENTFILE authority" [*SQL Anywhere Server - Database Administration*].
- **New WRITECLIENTFILE authority** The WRITECLIENTFILE authority allows a user to write to a file on a client computer. For example, to use the UNLOAD TABLE statement to unload data to a file on the client computer, the user needs WRITECLIENTFILE authority. See "WRITECLIENTFILE authority" [*SQL Anywhere Server - Database Administration*].
- **CREATE permission supported for dbspaces** SQL Anywhere now supports the CREATE ON permission for creating database object on the specified dbspace. The CREATE permission can be assigned to the user, or inherited through group membership. See "GRANT statement" [*SQL Anywhere Server - SQL Reference*] and "Understanding permissions" [*SQL Anywhere Server - Database Administration*].

Database utilities

Following is a list of enhancements made to database utilities in SQL Anywhere version 11.0.0.

- **Configuration file enhancement** You can now use an ampersand (&) in configuration files to indicate that the previous token is continued on the next line. See "Using configuration files" [*SQL Anywhere Server - Database Administration*].
- **Unload utility (dbunload) enhancements** The following enhancements have been made to dbunload:
  - A new option, -cp, has been added to allow you to cause dbunload to compress the data output file.
  - Previously, if specified the -ek, -ep, or -ea encryption options, without also specifying the -an or -ar reload options, an error was returned. Now, however, dbunload accepts the encryption options, and applies them to the output file it creates.

- The -g option now refreshes text indexes defined as MANUAL REFRESH. See "How to manage text indexes" [*SQL Anywhere Server - SQL Usage*].
- By default, text indexes defined as MANUAL REFRESH are not initialized as part of a reload. If you want to initialize these text indexes, you can specify the dbunload -g option.
- The -no option lets you unload object definitions in alphabetical order, grouped by object type. This can be useful for comparing the *reload.sql* files for databases.

See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

- **Validation utility (dbvalid) enhancements** Previously, the dbvalid utility validated all tables and materialized views by default. Dbvalid now also executes a VALIDATE DATABASE statement.

  When a database is started automatically by running the Validation utility, it is started in read-only mode. This prevents you from changing the database if it is being validated as part of a backup and recovery plan.

  See "Validation utility (dbvalid)" [*SQL Anywhere Server - Database Administration*].

- **Maintaining the generated *.sql* file if the Log Translation utility (dbtran) detects corruption** The Log Translation utility -k option lets you specify that you do not want a partial *.sql* file deleted if translating the log file fails because of corruption in the transaction log file. See "Log Translation utility (dbtran)" [*SQL Anywhere Server - Database Administration*].

Database options

Following is a list of enhancements made to database options in SQL Anywhere version 11.0.0.

- **allow_read_client_file option** This option controls whether to allow the reading of files on a client computer. See "allow_read_client_file option" [*SQL Anywhere Server - Database Administration*].
- **allow_write_client_file option** This option controls whether to allow the writing of files to a client computer. See "allow_write_client_file option" [*SQL Anywhere Server - Database Administration*].
- **login_procedure option** You can now signal an expired password error message. See "login_procedure option" [*SQL Anywhere Server - Database Administration*].
- **max_priority option** This option controls the maximum priority level for database connections. See "max_priority option" [*SQL Anywhere Server - Database Administration*].
- **priority option** This option controls the priority level at which requests from a connection are executed. See "priority option" [*SQL Anywhere Server - Database Administration*].
- **query_mem_timeout option** This option controls how long a request waits for a memory grant. See "query_mem_timeout option" [*SQL Anywhere Server - Database Administration*].

Database server options

Following is a list of enhancements made to database server options in SQL Anywhere version 11.0.0.

- **-es server option** In previous versions of SQL Anywhere, if the database server was started with the -ec option (to support transport layer security), but the list of allowed encryption protocols did not include NONE or SIMPLE, then the shared memory port was not started because it does not support transport layer security. This meant that all connections to the database server had to be made over TCP/IP using strong encryption.

  The -es server option instructs the database server to allow unencrypted connections over shared memory. See "-es dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-gb server option** The -gb server option for controlling the server process priority class is now supported on Unix, as well as Windows. See "-gb dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-im server option** You can run the database entirely in memory if the loss of all database operations is tolerable for your application. This feature is intended for situations where SQL Anywhere is intended to be used as a fast, temporary data-store, where data is inserted at a rapid rate. See "-im dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Reading from and writing to files on a client computer** The -sf server option now allows you to control the ability to read from, and write to, files on a client computer. See "-sf dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-um server option** The -um option allows you to connect to the *DBLauncher.app* instance, if it is running, and displays database server messages in a new window within *DBLauncher.app*. This option only applies to Mac OS X. See "-um dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Windows Performance Monitor options** The following server options have been added to further configure the Windows Performance Monitor:

  - **-ks option** Disables the creation of shared memory that the Performance Monitor uses to collect counter values from the database server. See "-ks dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

  - **-ksc option** Specifies the maximum number of connections that the Performance Monitor can monitor. See "-ksc dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

  - **-ksd option** Specifies the maximum number of databases that the Performance Monitor can monitor. See "-ksd dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Properties and Performance Monitor statistics

Following is a list of enhancements made to properties and Performance Monitor statistics in SQL Anywhere version 11.0.0.

- **New connection properties** The following connection properties have been added in this release:

  - allow_read_client_file
  - allow_write_client_file
  - AuthType
  - CacheReadWorkTable
  - ClientNodeAddress
  - DiskReadWorkTable
  - DiskSyncRead
  - DiskSyncWrite
  - DiskWaitRead
  - DiskWaitWrite
  - DiskWriteHint
  - DiskWriteHintPages
  - LockIndexID
  - LockRowID
  - max_priority
  - OSUser
  - priority
  - query_mem_timeout
  - QueryMemActiveCurr
  - QueryMemExtraAvail
  - QueryMemGrantFailed
  - QueryMemGrantGranted
  - QueryMemGrantWaiting
  - QueryMemGrantRequested
  - QueryMemWaited
  - ServerNodeAddress
  - ReadHint
  - ReadHintScatter

  For descriptions of these properties, see “Connection properties” [*SQL Anywhere Server - Database Administration*].

- **New database server properties** The following database server properties have been added in this release:

  - DiskRetryRead
  - DiskRetryReadScatter
  - DiskRetryWrite
  - EventTypeDesc
  - EventTypeName
  - HttpAddresses
  - HttpsAddresses
  - HttpNumActiveReq
  - HttpNumConnections
  - HttpNumSessions
  - HttpsNumActiveReq
  - HttpsNumConnections
  - MaxEventType
  - MaxRemoteCapability
  - MessageCategoryLimit
  - OptionWatchAction
  - OptionWatchList
  - QueryMemActiveCurr
  - QueryMemActiveEst
  - QueryMemActiveMax
  - QueryMemExtraAvail
  - QueryMemGrantBase
  - QueryMemGrantBaseMI
  - QueryMemGrantExtra
  - QueryMemGrantFailed
  - QueryMemGrantGranted
  - QueryMemGrantWaiting
  - QueryMemGrantRequested
  - QueryMemPages
  - QueryMemPercentOfCache
  - QueryMemWaited
  - ReadHintScatterLimit
  - RemoteCapability
  - StreamsUsed
  - TcpIpAddresses
  - WebClientLogFile
  - WebClientLogging

  For descriptions of these properties, see “Database server properties” [*SQL Anywhere Server - Database Administration*].

- **New database properties** The following database properties have been added in this release:

  - AlternateMirrorServerName
  - CacheReadWorkTable
  - DiskReadWorkTable
  - DiskRetryReadScatter
  - DiskSyncRead
  - DiskSyncWrite
  - DiskWaitRead
  - DiskWaitWrite
  - DiskWriteHint
  - DiskWriteHintPages
  - HasEndianSwapFix
  - MirrorMode
  - ReadHint
  - ReadHintScatter

  For descriptions of these properties, see "Database properties" [*SQL Anywhere Server - Database Administration*].

- **New Performance Monitor statistics** The following Performance Monitor statistics have been added in this release:

  - Cache Reads: Work Table
  - Disk Reads: Work Table

System procedures and functions

Following is a list of system procedure and function enhancements added in SQL Anywhere version 11.0.0.

- **sa_get_dtt_groupreads system procedure** The new sa_get_dtt_groupreads system procedure allows you to estimate the cost of issuing group reads on the database server. See "sa_get_dtt_groupreads system procedure" [*SQL Anywhere Server - SQL Reference*].

- **PROPERTY_NAME function enhancement** Now returns the name of the property with the supplied property ID for the specified connection level. See "PROPERTY_NAME function [System]" [*SQL Anywhere Server - SQL Reference*].

- **READ_CLIENT_FILE function** The new READ_CLIENT_FILE function reads data from the specified file on the client computer. See "READ_CLIENT_FILE function [String]" [*SQL Anywhere Server - SQL Reference*].

- **WRITE_CLIENT_FILE function** The new WRITE_CLIENT_FILE function writes data to the specified file on the client computer. See "WRITE_CLIENT_FILE function [String]" [*SQL Anywhere Server - SQL Reference*].

- **REGEXP_SUBSTR function** The new REGEXP_SUBSTR function allows you to search for a substring within a string. This new function takes a regular expression as an argument. See "REGEXP_SUBSTR function [String]" [*SQL Anywhere Server - SQL Reference*].

- **sa_char_terms system procedure** The new sa_char_terms system procedure breaks a CHAR string into terms and returns every term together with its position. See "sa_char_terms system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_nchar_terms system procedure** The new sa_nchar_terms system procedure breaks an NCHAR string into terms and returns every term together with its position. See "sa_nchar_terms system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_refresh_text_indexes system procedure** The new sa_refresh_text_indexes system procedure refreshes all text indexes that are defined as MANUAL REFRESH or AUTO REFRESH. See "sa_refresh_text_indexes system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_text_index_stats system procedure** The new sa_text_index_stats system procedure returns statistical information for all text indexes in the database, including the last refresh time and size of pending changes. See "sa_text_index_stats system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_text_index_vocab system procedure** The new sa_text_index_vocab system procedure lists all terms that appear in a text index, and the total number of indexed values that each term appears in. See "sa_text_index_vocab system procedure" [*SQL Anywhere Server - SQL Reference*].

  Two new system procedures, sa_internal_text_index_vocab and sa_internal_text_index_postings have also been added, but are only for use by the sa_text_index_vocab system procedure.

- **sa_text_index_postings system procedure** This new system procedure is for internal use only.

- **sa_text_index_handles system procedure** This new system procedure is for internal use only.

- **sa_get_user_status system procedure** The new sa_get_user_status system procedure allows you to determine a user's current login status. See "sa_get_user_status system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Running procedures and functions as invoker** When creating a procedure or function, you can now specify whether the procedure or function runs as though it was called by the user calling it (invoker), or by the user who created it (definer). To specify this, use the SQL SECURITY clause of the CREATE PROCEDURE or CREATE FUNCTION statement. See "CREATE FUNCTION statement" [*SQL Anywhere Server - SQL Reference*] and "CREATE PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*].

  This change also applies to external procedures and functions.

- **sa_disk_free_space system procedure** The sa_disk_free_space system procedure now returns a new column, total_space, indicating the total amount of disk space available on the drive where the dbspace resides. For databases created on versions of SQL Anywhere before 11.0.0, the total_space column is not returned until the database is upgraded. See "sa_disk_free_space system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_external_library_unload system procedure** A new system procedure, sa_external_library_unload, has been added to allow you to unload external libraries that are not in use. See "sa_external_library_unload system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_index_density system procedure now returns skew** The sa_index_density system procedure has been enhanced to return the amount of skew present in the index. A high degree of skew can impact performance compared to a well balanced index. See "Reduce index fragmentation and skew" [*SQL Anywhere Server - SQL Usage*] and "sa_index_density system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_materialized_view_info system procedure enhancements** Information in the Status column returned by sa_materialized_view_info has been split into two columns, Status and DataStatus. The Status now returns information on whether the view is enabled or disabled. The new DataStatus column returns information about whether there is data in the view, and the freshness of the data. An additional column, RefreshType, has been added to indicate whether the view is a manual view or an immediate view. See "sa_materialized_view_info system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_materialized_view_can_be_immediate system procedure** Newly created materialized views are manual views by default, but can be altered to become immediate views, providing they do not violate any of the restrictions for immediate views. The new sa_materialized_view_can_be_immediate system procedure allows you to test whether a manual view can be changed to an immediate view. See "sa_materialized_view_can_be_immediate system procedure" [*SQL Anywhere Server - SQL Reference*] and "Additional restrictions for immediate views" [*SQL Anywhere Server - SQL Usage*].

- **sa_post_login_procedure system procedure** A new system procedure has been added to allow you to determine if a warning should be issued when a user's password is about to expire. See "sa_post_login_procedure system procedure" [*SQL Anywhere Server - SQL Reference*].

- **EVENT_PARAMETER function enhancement** The EVENT_PARAMETER function now supports abnormal as a DisconnectReason. This new reason indicates that a disconnect occurred either as a result of the client application shutting down abnormally before disconnecting from the database, or as a result of a communication failure between the client and server computers. See "EVENT_PARAMETER function [System]" [*SQL Anywhere Server - SQL Reference*].

- **sa_server_option system procedure enhancements** Two new properties, OptionWatchList and OptionWatchAction, have been added to the sa_server_option system procedure. You can use these properties to monitor when an attempt is made to change a database option setting, and specify the action to take. See "Monitoring option settings" [*SQL Anywhere Server - Database Administration*] and "sa_server_option system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_db_properties system procedure enhancement** The sa_db_properties system procedure now returns valid properties that have NULL values. See "sa_db_properties system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_conn_properties system procedure enhancement** The sa_conn_properties system procedure now returns valid properties that have NULL values. See "sa_conn_properties system procedure" [*SQL Anywhere Server - SQL Reference*].

SQL statements

Following is a list of SQL enhancements introduced in SQL Anywhere version 11.0.0.

- **New CALIBRATE GROUP READ clause, ALTER DATABASE statement** The new CALIBRATE GROUP READ clause of the ALTER DATABASE statement allows you to perform group read calibration on the temporary dbspace. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **New CHECK clause, CREATE MATERIALIZED VIEW statement** The new CHECK clause of the CREATE MATERIALIZED VIEW statement allows you to validate the statement before creating the view. See "CREATE MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*].

- **New RECOMPILE clause, ALTER FUNCTION statement** A new clause, RECOMPILE, has been added to the ALTER FUNCTION statement to allow you to recompile a user-defined function. See "ALTER FUNCTION statement" [*SQL Anywhere Server - SQL Reference*].

- **New RECOMPILE clause, ALTER PROCEDURE statement** A new clause, RECOMPILE, has been added to the ALTER PROCEDURE statement to allow you to recompile a stored procedure. See "ALTER PROCEDURE statement" [*SQL Anywhere Server - SQL Reference*].

- **New REFRESH clause, ALTER MATERIALIZED VIEW statement** A new clause, REFRESH, has been added to the ALTER MATERIALIZED VIEW statement to allow you to specify the refresh type for the materialized view. See "ALTER MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*].

- **LOAD TABLE statement enhancements in support of recovery and mirroring** The following clauses have been added to the LOAD TABLE statement in support of recovery and mirroring:

  - **WITH CONTENT LOGGING clause** The WITH CONTENT LOGGING clause instructs the database server to record the contents of the data source in the transaction log. The data is recorded in small chunks as the input is processed by LOAD TABLE. These chunks can be reconstituted into rows by a mirroring database, or when recovering from the transaction log. The WITH CONTENT LOGGING clause can be beneficial when it is not desirable to maintain the original data files for later recovery. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

  - **WITH ROW LOGGING clause** The WITH ROW LOGGING clause instructs the database server to record, as a series of INSERT statements, all the rows being loaded. This level is ideal for databases involved in synchronization, as well as in situations where the table being loaded into contains non-deterministic values, such as computed columns, or CURRENT TIMESTAMP defaults. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

  - **WITH FILE NAME LOGGING clause** The WITH FILE NAME LOGGING clause instructs the database server to record only the LOAD TABLE statement. This is the default behavior and is consistent with the logging behavior in previous versions of SQL Anywhere. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **New client file loading and unloading clauses** The enhancements have been made to the LOAD TABLE and UNLOAD TABLE statements in support of the new client file loading/unloading feature:
  - **New USING CLIENT FILE clause for the LOAD TABLE statement** Allows you to load a table using data in a file located on the client computer. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].
  - **New INTO CLIENT FILE clause for the UNLOAD TABLE statement** Allows you to specify a file on the client computer to unload data into. See "UNLOAD statement" [*SQL Anywhere Server - SQL Reference*].
- **New login policy statements** The following statements have been added in support of the new login policy feature:
  - "CREATE LOGIN POLICY statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER LOGIN POLICY statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP LOGIN POLICY statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE USER statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER USER statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP USER statement" [*SQL Anywhere Server - SQL Reference*]
- **New full text search statements and clauses** The following statements have been added in support of the new full text search feature:
  - **New CONTAINS search condition** The CONTAINS search condition is used to check a specified list of columns for the existence of a specified list of terms or phrases. The CONTAINS search condition returns either TRUE or FALSE. When searching for multiple terms or phrases, you can combine them with various Boolean operators. See "CONTAINS search condition" [*SQL Anywhere Server - SQL Reference*].
  - **New CONTAINS clause in the FROM clause of a SELECT statement** The CONTAINS clause is specified in the FROM clause of a SELECT statement and works much like the CONTAINS search condition but also returns a score for each matching column, and an overall score for each matching row. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].
  - **CREATE TEXT CONFIGURATION statement** This statement creates a text configuration object. A text configuration object is a set of configuration settings that control characteristics of a text index. See "CREATE TEXT CONFIGURATION statement" [*SQL Anywhere Server - SQL Reference*].
  - **ALTER TEXT CONFIGURATION statement** This statement alters a text configuration object. See "ALTER TEXT CONFIGURATION statement" [*SQL Anywhere Server - SQL Reference*].
  - **DROP TEXT CONFIGURATION statement** This statement drops a text configuration object. See "DROP TEXT CONFIGURATION statement" [*SQL Anywhere Server - SQL Reference*].

- **CREATE TEXT INDEX statement** This statement creates a text index. A text index stores complete positional information for every instance of every term in every indexed column. See "CREATE TEXT INDEX statement" [*SQL Anywhere Server - SQL Reference*].
- **ALTER TEXT INDEX statement** This statement alters a text index. See "ALTER TEXT INDEX statement" [*SQL Anywhere Server - SQL Reference*].
- **DROP TEXT INDEX statement** This statement removes a text index from the database. See "DROP TEXT INDEX statement" [*SQL Anywhere Server - SQL Reference*].
- **REFRESH TEXT INDEX statement** This statement refreshes a text index. See "REFRESH TEXT INDEX statement" [*SQL Anywhere Server - SQL Reference*].
- **TRUNCATE TEXT INDEX statement** This statement truncates the data from a text index. See "TRUNCATE TEXT INDEX statement" [*SQL Anywhere Server - SQL Reference*].

- **ALTER EVENT statement enhancement** You can now hide the definition for an event handler using the ALTER EVENT ... SET HIDDEN statement. This statement results in the obfuscation of the event handler definition stored in the action column of the ISYSEVENT system table. See "ALTER EVENT statement" [*SQL Anywhere Server - SQL Reference*].
- **BEGIN SNAPSHOT statement** The BEGIN SNAPSHOT statement lets you control when a snapshot starts for snapshot isolation. See "BEGIN SNAPSHOT statement" [*SQL Anywhere Server - SQL Reference*].
- **CASE statement and CASE expression enhancements** For improved compatibility, CASE statements and CASE expressions are now permitted to end with either END or END CASE. See "CASE statement" [*SQL Anywhere Server - SQL Reference*] and "CASE expressions" [*SQL Anywhere Server - SQL Reference*].
- **COMMENT statement enhancements** You can now add comments to the login policies table and to dbspaces. See:
  - "COMMENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "Managing login policies" [*SQL Anywhere Server - Database Administration*]
  - "Using additional dbspaces" [*SQL Anywhere Server - Database Administration*]
- **CREATE MATERIALIZED VIEW statement enhancement** You can now create a materialized view that is refreshed whenever underlying data changes, using the new IMMEDIATE REFRESH clause of the CREATE MATERIALIZED VIEW statement. See "CREATE MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*].
- **DESCRIBE statement enhancement** The Interactive SQL DESCRIBE statement now allows you to obtain information about the database or database server that Interactive SQL is connected to. See "DESCRIBE statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].
- **IF statement and IF expression enhancements** For improved compatibility, IF statements and IF expressions are now permitted to end with either ENDIF or END IF. See "IF statement" [*SQL Anywhere Server - SQL Reference*] and "IF expressions" [*SQL Anywhere Server - SQL Reference*].

- **LOAD TABLE statement enhancements** When using the LOAD TABLE statement, you can now specify whether the data in the input file is compressed, and/or encrypted, using the new COMPRESSED or ENCRYPTED clauses. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **SELECT statement enhancements**

  - **Enhancements to INDEX clause** When specifying an index hint using the INDEX clause, you can now specify up to four indexes that the database server must use. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

  - **New INDEX ONLY clause** When specifying an index hint using the INDEX clause, you can optionally specify the INDEX ONLY clause to control whether the database server performs an index-only retrieval (that is, only index data is used to satisfy the query). See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

  - **New CROSS APPLY and OUTER APPLY clauses** The SELECT statement has been extended to support apply expressions (specifically, the CROSS APPLY and OUTER APPLY clauses) in the FROM clause. An apply expression is an easy way to specify joins where the right side is dependent upon the left. For example, you can use an apply expression to evaluate a procedure or derived table once for each row in a table expression. See "Joins resulting from apply expressions" [*SQL Anywhere Server - SQL Usage*] and "FROM clause" [*SQL Anywhere Server - SQL Reference*].

  - **New OPENSTRING clause** Using the new OPENSTRING clause, you can now use a SELECT statement to query data in a file. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

- **Specify an owner when creating, altering, dropping, or commenting on events** The CREATE EVENT, ALTER EVENT, DROP EVENT, and COMMENT ON EVENT statements now allow you to optionally specify the owner. See:

  - "CREATE EVENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER EVENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP EVENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "COMMENT statement" [*SQL Anywhere Server - SQL Reference*]

- **UNLOAD statement enhancements** When using the UNLOAD statement, you can now specify whether to compress and/or encrypt the data that is being unloaded by specifying the COMPRESSED or ENCRYPTED clauses, respectively. See "UNLOAD statement" [*SQL Anywhere Server - SQL Reference*].

  Files compressed or encrypted using these clauses can only be loaded (for example, using LOAD TABLE) by SQL Anywhere 11.0.0 database servers. Files compressed or encrypted using other tools are not usable by SQL Anywhere.

- **UPDATE statement enhancement** For both search and positioned updates, you can now use the SET clause to set the column value to its default value. See "UPDATE statement" [*SQL Anywhere*

Server - SQL Reference] and "UPDATE (positioned) statement [ESQL] [SP]" [*SQL Anywhere Server - SQL Reference*].

- **Extension to the OPTION clause** The OPTION clause for the INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements can now override the setting of the user_estimates database option. See:
  - "INSERT statement" [*SQL Anywhere Server - SQL Reference*]
  - "UPDATE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DELETE statement" [*SQL Anywhere Server - SQL Reference*]
  - "SELECT statement" [*SQL Anywhere Server - SQL Reference*]
  - "UNION statement" [*SQL Anywhere Server - SQL Reference*]
  - "EXCEPT statement" [*SQL Anywhere Server - SQL Reference*]
  - "INTERSECT statement" [*SQL Anywhere Server - SQL Reference*]

Programming interfaces

Following is a list of enhancements to programming interfaces introduced in SQL Anywhere version 11.0.0.

- **New SQL Anywhere C API** The SQL Anywhere C application programming interface (API) simplifies the creation of C and C++ wrapper drivers for several interpreted programming languages, including PHP, Perl, Python, and Ruby. The SQL Anywhere C API is layered on top of the DBLIB library and it was implemented with Embedded SQL.

  Although it is not a replacement for DBLIB, the SQL Anywhere C API simplifies the creation of applications using C and C++. You do not need an advanced knowledge of embedded SQL to use the SQL Anywhere C API. See "SQL Anywhere C API support" [*SQL Anywhere Server - Programming*].

- **New Python Database API (sqlanydb)** The new Python Database API (sqlanydb) provides access to SQL Anywhere databases from scripts written in Python. The sqlanydb module implements, with extensions, the Python Database API specification, version 2.0. See "Python support" [*SQL Anywhere Server - Programming*].

- **External environments** SQL Anywhere now includes support for six external runtime environments: Java, Perl, PHP, CLR, embedded SQL, and ODBC. SQL Anywhere has had the ability to call compiled native functions written in C or C++ for some time. However, when these procedures are run by the server, the dynamic link library or shared object has always been loaded by the database server and the calls out to the native functions have always been made by the database server. The risk here is that if the native function causes a fault, then the database server will crash. Running compiled native functions outside the database server, in an external environment, eliminates these risks to the server. See "SQL Anywhere external environment support" [*SQL Anywhere Server - Programming*].

  A database upgrade is required to take advantage of this new feature. See "Upgrading SQL Anywhere" on page 304.

- **PHP external environment support** SQL Anywhere 11.0.0 includes a variety of pre-built binaries for various PHP versions including 5.1.1 through 5.1.6 and 5.2.0 through 5.2.6. If you have any one of these versions already installed on your server computer, then you should use the SQL Anywhere pre-built binaries instead of building the PHP external environment yourself. Note that, for

Linux and Solaris, both 32-bit and 64-bit versions of these binaries are provided. For Windows and other systems, only 32-bit versions are provided.

If you have a different PHP version installed than the ones listed above, then you must build the software, or switch your PHP version to one that matches a SQL Anywhere prebuilt version. For instructions on building the SQL Anywhere PHP module, see "SQL Anywhere PHP extension" [*SQL Anywhere Server - Programming*].

- **Perl external environment support** It is very important that you update your version of the SQL Anywhere Perl DBD driver before you try to use the Perl external environment. If you do not update your Perl DBD driver, then server-side Perl will not work.

  Also, unlike PHP, SQL Anywhere does not include pre-built binaries for various versions of Perl. Source code for the SQL Anywhere Perl DBD driver is located in *install-dir\SDK\perl*. For instructions on building the SQL Anywhere Perl DBD driver, see "Perl DBI support" [*SQL Anywhere Server - Programming*].

- **Web server support for UTF-8 URLs** Previously, the web server decoded percent-encoded (%encoded) data within the request URL (or application/x-www-form-urlencoded data within the body of the request) into the database character set. Now, the contents of percent-encoded (%encoded) data is tested for UTF-8 sequences and converted to the database character set on a maximal extent basis. Any encoded data that is not UTF-8 is decoded and treated as if it is already in the database character set.

  Client HTTP applications should send percent-encoded (%encoded) UTF-8 data exclusively. Note that ASCII is represented in UTF-8 as is. For example, a space is encoded as %20.

- **New client callback API** A new client callback API has been added in support of the new client-side loading and unloading of data features. For embedded SQL, see DB_CALLBACK_VALIDATE_FILE_TRANSFER in "db_register_a_callback function" [*SQL Anywhere Server - Programming*]. For ODBC, see SA_REGISTER_VALIDATE_FILE_TRANSFER_CALLBACK in "SQLSetConnectAttr extended connection attributes" [*SQL Anywhere Server - Programming*].

- **SQL_ATTR_CONNECTION_DEAD promptly detects dead connection** Using ODBC's SQLGetConnectAttr call to get the SQL_ATTR_CONNECTION_DEAD attribute now gets the value SQL_CD_TRUE if the connection has been dropped even if no request has been made to the server since the connection was dropped. Determining if the connection has been dropped is done without making a request to the server, and the dropped connection is detected within a few seconds. The connection can be dropped for several reasons, for example, on an idle timeout. Before this change, SQL_ATTR_CONNECTION_DEAD only got the value SQL_CD_TRUE if the connection was disconnected or if ODBC driver made a request to the server (by calling SQLExecDirect for example) after the connection was dropped. See "Getting connection attributes" [*SQL Anywhere Server - Programming*].

- **JDBC Driver now supports ResultSet.getBlob().getBinaryStream()** The iAnywhere JDBC Driver currently supports the ResultSet.getBlob() method even though this method is optional in the JDBC specification. Support has been added for the optional ResultSet.getBlob().getBinaryStream() method. See "JDBC 3.0/4.0 API support" [*SQL Anywhere Server - Programming*].

- **iAnywhere JDBC driver now accepts jdbc:ianywhere as URL header in addition to jdbc:odbc** Previously, applications using the URL header jdbc:odbc could be reasonably certain that the JDBC Driver Manager would use the iAnywhere JDBC driver for making connections using this URL. However, recent versions of the Java VM have started to register the Sun JDBC-ODBC bridge as a JDBC driver, and since the Sun JDBC-ODBC bridge also accepts URLs beginning with jdbc:odbc, the chance of an application getting the Sun JDBC-ODBC bridge instead of the iAnywhere JDBC driver is quite high. To guarantee that the JDBC Driver Manager uses the iAnywhere JDBC driver instead of the Sun JDBC-ODBC bridge, the application should use the URL header jdbc:ianywhere instead. See "Connecting from a JDBC client application" [*SQL Anywhere Server - Programming*].

- **ODBC driver manager now accepts driver=iAnywhere Solutions 11 - Oracle** The Unix ODBC driver manager now accepts driver=iAnywhere Solutions 11 - Oracle, and it loads the threaded iAnywhere ODBC driver for Oracle if the application is threaded. It does not load the driver if the application is non-threaded because the non-threaded iAnywhere ODBC driver for Oracle is not supported. See "iAnywhere Solutions 12 - Oracle ODBC driver" [*MobiLink - Server Administration*].

- **ODBC driver manager now accepts driver=UltraLite 11** The Unix ODBC driver manager already accepts driver=SQL Anywhere 10 and loads the SQL Anywhere ODBC driver (either threaded or non-threaded, depending on the application). The Unix ODBC driver manager now accepts driver=SQL Anywhere 11 and driver=UltraLite 11. For the UltraLite driver, the driver manager *only* loads the threaded version of the UltraLite ODBC driver because only the threaded version exists.

- **TDS connections enhancement** The SQL Anywhere database server now allows TDS connections to the default database, even when the Open Client login server name does not match the name of the default database, if the connection string does not involve starting a database (that is, there is no `DBF=...`) and if the database server is only running one database.

- **Administration Tool launchers now easier to redeploy** The launcher executables for the database tools (Sybase Central, DBISQL, DBConsole, ML Monitor) are now easier to redeploy. Registry entries and a set directory structure for the location of the JAR files are no longer required. Each executable needs to have an *.ini* file in the same directory (with the same name as the executable file) containing the details on how to load the tool. See "Deploying administration tools" [*SQL Anywhere Server - Programming*].

- **SQL Anywhere .NET Data Provider now supports distributed transaction enlistment** The .NET 2.0 framework introduced a new namespace System.Transactions, which contains classes for writing transactional applications. Client applications can create and participate in distributed transactions with one or multiple participants. Client applications can implicitly create transactions using the TransactionScope class. The connection object can detect the existence of an ambient transaction created by the TransactionScope and automatically enlist. Client applications can also create a CommittableTransaction and call the EnlistTransaction method to enlist.

  This feature is supported by the SQL Anywhere .NET 2.0 Data Provider. Distributed transaction has significant performance overhead. It is recommended that you use database transactions for non-distributed transactions. See "Transaction processing" [*SQL Anywhere Server - Programming*].

- **SQL Anywhere .NET Data Provider now supports named parameters** The SQL Anywhere provider now supports named parameters in SACommand. If the user specifies all parameter names,

the provider maps the parameter values when the command is executed. When you use named parameters, the order of parameters is not required to match the order of host variables.

```
SACommand cmd = new SACommand(
    "UPDATE MyTable SET name = :name WHERE id = :id", conn );

SAParameter p1 = new SAParameter(
    "id", SADbType.Integer );
p1.Direction = ParameterDirection.Input;
p1.Value = 1;
cmd.Parameters.Add( p1 );

SAParameter p2 = new SAParameter(
    "name", SADbType.Char, 40 );
p2.Direction = ParameterDirection.Input;
p2.Value = "asdasd";
cmd.Parameters.Add( p2 );

cmd.ExecuteNonQuery();
```

- **Web services enhancements** The following web services enhancements have been made in this release:

  - **Extending web client service procedures of type HTTP:POST to allow a user-defined body** The TYPE clause of the CREATE PROCEDURE and CREATE FUNCTION statements has been extended to allow the specification of a mime type. See "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*] or "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*].

  - **Extending web service client procedures to support the PUT, DELETE, and HEAD HTTP methods** Web service client procedures and functions now support the PUT, DELETE and HEAD HTTP methods. The TYPE clause of the CREATE PROCEDURE and CREATE FUNCTION statements has been extended to support these methods. Similar to the POST method, PUT requires a content-type extension within the type clause and only a single (non-substitution) parameter is permitted. See "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*], "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*], and "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*].

  - **sa_http_php_page and sa_http_php_page_interpreted system procedures** The new web service system procedures sa_http_php_page and sa_http_php_page_interpreted return the result of passing a PHP script through a PHP interpreter. See "sa_http_php_page system procedure" [*SQL Anywhere Server - SQL Reference*] and "sa_http_php_page_interpreted system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **HTTP_BODY system function** A new web service function has been added. The HTTP_BODY function returns the body of the HTTP request in binary form. See "HTTP_BODY function [HTTP]" [*SQL Anywhere Server - SQL Reference*].

  - **WSDLC support for generating web service client SOAP procedures** In addition to generating QAnywhere client-side SOAP interfaces for C# and JAVA, WSDLC now supports the generation of SQL SOAP (web service) client procedures for SQL Anywhere. WSDLC reads a WSDL1.1 compliant URL or file and generates procedures (or functions) with appropriate

parameters and clauses that map to respective SOAP operations listed within the WSDL. The generated SQL statements are written to a SQL file. See "iAnywhere WSDL compiler utility (wsdlc)" [*SQL Anywhere Server - Programming*].

- **HTTP SOAP services defined with a FORMAT clause may be further qualified with EXPLICIT OFF or ON** When creating an HTTP SOAP service, the default for the FORMAT clause is EXPLICIT ON. This means that the WSDL generated by a DISH service specifies explicit names and data types for each column returned within a result set. This allows SOAP client toolkits to automatically generate client-side objects and interfaces that represent the result set providing native access to the column values. Before this feature, column values could only be accessed as abstract XML data elements. That behavior can still be achieved by specifying EXPLICIT OFF.

  For more information on how to define an EXPLICIT response object or the generic SimpleDataset, see "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*] and "Tutorial: Using JAX-WS to access a SOAP/DISH web service" [*SQL Anywhere Server - Programming*].

- **Support for JSON web services** SQL Anywhere now supports web services that return JSON-formatted responses. See "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*].

- **Logging web service clients** The database server now supports logging web service client connections to an output file. You can specify the -zoc server option or use the WebClientLogFile and WebClientLogging properties with the sa_server_option system procedure to control logging and specify the location of the web service client log file. You can also disable the use of this feature with the -sf server option. See:

  - "-zoc dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]
  - "-sf dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]
  - "sa_server_option system procedure" [*SQL Anywhere Server - SQL Reference*]

Windows Mobile enhancements

Following is a list of Windows Mobile enhancements introduced in SQL Anywhere version 11.0.0.

- **-gss server option supported** On Windows CE 4 (Pocket PC 2003) and later, you can use the -gss server option to specify the default stack size for internal execution threads. See "-gss dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Checksums enabled by default** When a database is running on Windows Mobile, the database server enables checksums automatically, regardless of whether checksums were enabled for the database. You must upgrade existing databases or create a new database to use this feature. See "Using checksums to detect corruption" [*SQL Anywhere Server - Database Administration*].

Unix/Linux enhancements

Following is a list of Unix and Linux enhancements introduced in SQL Anywhere version 11.0.0.

- **Controlling the permissions for temporary files** In previous releases, temporary files created by the database server and client were created with global read, write and execute permissions. You can control the permissions for temporary files by setting the SATMP environment variable to a directory with the desired permissions. See "SATMP environment variable" [*SQL Anywhere Server - Database Administration*].

- **SELinux support** SELinux policies control an application's access to system resources. You can use the default policy on Red Hat Enterprise Linux 5 with SQL Anywhere, but SQL Anywhere is not secured when it is run this way. SQL Anywhere now includes a policy that secures it on Red Hat Enterprise Linux 5. You must compile and install the policy for it to work. The policy source code is provided as part of your SQL Anywhere installation.

  For information about compiling and installing the SQL Anywhere SELinux policy, see *install-dir/selinux/readme*.

- **Applications menu items on Linux** When installing SQL Anywhere 11 on Linux, you can choose to create **Applications** menu items.

Mac OS X enhancements

- **Encryption now supported on Mac OS X** RSA communication encryption is now supported by both the database server and clients on Mac OS X. For information about using strong encryption, see "Transport-layer security" [*SQL Anywhere Server - Database Administration*].

- **HTTPS now supported on Mac OS X** HTTPS communications are now supported by the database server on Mac OS X. For information about using HTTPS, see "-xs dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Miscellaneous

Following is a list of miscellaneous enhancements introduced in SQL Anywhere version 11.0.0.

- **Altering invalid views** Previously a regular view with INVALID status could not be altered, requiring you to drop the view and recreate it. Now, you can alter an invalid view to change the definition so that it is no longer invalid.

- **Support added for big-endian and little endian UTF-16 encodings** SQL Anywhere now supports both big-endian and little endian UTF-16 encoding on all platforms, regardless of the endianness of the platform. You can use UTF-16 encoding in the LOAD TABLE and UNLOAD statements and with the CSCONVERT function. However, you cannot use UTF-16 encoding as the encoding for a connection or database. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*] and "UNLOAD statement" [*SQL Anywhere Server - SQL Reference*].

  A database upgrade is required to take advantage of this feature. See "Upgrading SQL Anywhere" on page 304.

- **Index performance enhancements** Index performance has been improved, especially when you are operating with a full cache. To benefit from the index performance enhancements, you must

rebuild your indexes. The easiest way to do this is to rebuild the database. After rebuilding, you may find that your database file is much smaller. This is normal and should not be a cause for concern.

- **INLINE and PREFIX settings now respected for compressed columns** Previously, the INLINE and PREFIX settings specified for a column were ignored and treated as 0 if the column was compressed. Now, the settings for the column are respected, even if the column is compressed. See "Storing BLOBs" [*SQL Anywhere Server - Database Administration*] and "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **Host variables now allowed in batches** References to host variables are now allowed within batches, with some restrictions. See "Introduction to batches" [*SQL Anywhere Server - SQL Usage*].

- **Enhancements to the InList algorithm** Previously, the InList algorithm was used by the optimizer only if all the elements of the IN list were either constant values or could be evaluated at optimization time to a constant value. Now, the IN list predicate can contain values that are evaluated only at open time (such as CURRENT DATE, CURRENT TIMESTAMP, or non-deterministic system and user-defined functions), as well as values that are constant within one execution of a query block (outer references). See "InList algorithm (IN)" [*SQL Anywhere Server - SQL Usage*].

- **Plan caching for simple DML statements** Plan caching has been extended to include SELECT statements that qualify for query bypass (simple statements). See "Plan caching" [*SQL Anywhere Server - SQL Usage*].

- **Size of new databases reduced** The following system table columns are now compressed to reduce the size of new (empty) databases by approximately 200 KB. This is beneficial when creating databases for use on Windows Mobile.

  - ISYSEVENT.action
  - ISYSJARCOMPONENT.contents
  - ISYSPROCEDURE.proc_defn
  - ISYSSOURCE.source
  - ISYSTEXTCONFIG.char_stoplist
  - ISYSTEXTCONFIG.nchar_stoplist
  - ISYSTRIGGER.trigger_defn
  - ISYSVIEW.view_def

- **Increased default and minimum packet size** The default packet size has been increased to 7300 bytes on all operating systems except Windows Mobile. On Windows Mobile, the default continues to be 1460 bytes. The minimum packet size has been increased to 500 bytes. See "CommBufferSize (CBSIZE) connection parameter" [*SQL Anywhere Server - Database Administration*] and "-p dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **New ODBC classes supported for remote data access** Support for the following ODBC classes has been added:

  - msaccessodbc
  - mysqlodbc
  - ulodbc
  - adsodbc

For more information, see "ODBC-based server classes" [*SQL Anywhere Server - SQL Usage*].

Migrating Access databases
If you previously used the SQL Anywhere for MS Access Migration utility (upsize tool) to migrate Microsoft Access databases to SQL Anywhere, you can now use the msaccessodbc class.

- **Database server messages enhancements** Messages from the database server now have a category and severity assigned to them. You can access this information using the sa_server_messages system procedure, and you can configure the number of messages maintained with the MessageCategoryLimit property. See "sa_server_messages system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New VALIDATE_COMPLETE parameter for a_validate_type enumeration** The a_validate_type enumeration has a new parameter, VALIDATE_COMPLETE for performing all possible validations on the database. See "a_validate_db structure" [*SQL Anywhere Server - Programming*].

- **External unload enhancements** When you perform an external unload of a database, the beginning of the *reload.sql* that is generated now contains a commented CREATE DATABASE statement. This statement can be used to create a database that is equivalent to the one being unloaded.

  If the unloaded database was created with version 9 or earlier of SQL Anywhere and had a custom collation, the COLLATION clause appears as follows:

  ```
  COLLATION collation-label DEFINITION collation-definition
  ```

  where *collation-definition* is a string that specifies the custom collation.

  If the unloaded database was created with strong encryption, the value of the KEY clause in the CREATE DATABASE statement appears as three question marks (???).

  For more information, see "Internal versus external unloads and reloads" [*SQL Anywhere Server - Database Administration*].

- **New SQL Anywhere Extension Agent OIDs** The following OIDs have been added in this release:

  - saAgent.saRestart
  - saAgent.saInifile

  For more information, see "SQL Anywhere MIB reference" [*SQL Anywhere Server - Database Administration*].

- **Deadlock system event** The Deadlock system event fires whenever a deadlock occurs. The event handler can use the sa_report_deadlocks procedure to obtain information about the conditions that led to the deadlock. You must upgrade existing databases if you want to use the Deadlock system event. See "Understanding system events" [*SQL Anywhere Server - Database Administration*].

- **Increased database limits** Several SQL Anywhere database limits have been increased. See "SQL Anywhere size and number limitations" [*SQL Anywhere Server - Database Administration*].

- **Changes to execution plans** Long plans generated by the optimizer now display the following entries related to the overall plan:
  - **Costed Best Plan** Number of different best access plans found by the optimizer.
  - **Costed Plans** Number of different access plans considered by the optimizer.
  - **Optimization Time** The time spent optimizing the query.

  See "Execution plan abbreviations" [*SQL Anywhere Server - SQL Usage*].

  Graphical plans now display the following entries:
  - **Costed Best Plan** Located in the **Optimizer Statistics** section of the root node, this entry provides the number of different best access plans found by the optimizer.
  - **Costed Plans** Located in the **Optimizer Statistics** section of the root node, this entry provides the number of different access plans considered by the optimizer.
  - **Optimization Time** Located in the **Optimizer Statistics** section of the root node, this entry provides the time spent optimizing the query.
  - **FirstRowRunTime** Located in any **Node Statistics** section, this entry provides the time to fetch the first row.
  - **Joins considered** Located in the **Advanced Details** section of any join operator, this entry lists all join operators considered by the optimizer during the optimization process for the subtree on the right-hand side of the join operator.
  - **Prefilter predicates** Located in a new scan node section in the **Details** pane, this entry lists all predicates that are evaluated before the scan is started.
  - **Scan predicates** Located in a scan node section in the **Details** pane, this entry lists the predicates that are evaluated as columns that are fetched from the row. If a scan predicate rejects a row, further columns are not read. Scan predicates are simple, single column predicates such as `T.x <= 3` or `T.x IS NULL`.
  - **Post scan predicates** Located in a new scan node section in the **Details** pane, this entry lists the predicates that are evaluated immediately after a row has been read from the table page. Post scan predicates can refer to multiple columns and can use functions or arithmetic.
  - **Residual predicates** Located in a new scan node section in the **Details** pane, this entry lists predicates that are evaluated after a set of rows has been fetched into memory. Residual predicates usually contain complex operations such as subqueries or user-defined functions and can not be evaluated as scan predicates or post scan predicates.
  - **Indexes considered** Located in the **Advanced Details** pane, this entry lists all the index or table scans considered by the optimizer during the optimization process for the table referenced by this scan operator. The format of each item in the list is similar to the details listed for a scan operator used in the access plan in the **Details** pane.

- **Primary Key Table** Located in the **Index** section of an index scan operator, this entry provides the primary key table name.
- **Primary Key Table Estimated Rows** Located in the **Index** section of an index scan operator, this entry provides the number of rows in the primary key table.
- **Primary Key Column** Located in the **Index** section of an index scan operator, this entry provides the names of the primary key columns.
- **Sequential Transitions** Located in the **Index** section of an index scan operator, this entry provides the statistics kept for each physical index indicating how clustered the index is.
- **Random Transitions** Located in the **Index** section of an index scan operator, this entry provides the statistics kept for each physical index indicating how clustered the index is.
- **Key Values** Located in the **Index** section of an index scan operator, this entry provides the number of unique entries in the index.

SQL Anywhere behavior changes

Following is a list of behavior changes to SQL Anywhere introduced in version 11.0.0, grouped by category.

- **Catalog changes** The following table contains the changes to the catalog for version 11.0.0.

  You must upgrade your database to get these changes. See "Upgrading SQL Anywhere" on page 304.

Table name and/or view name	Description of change	Additional information
ISYSTAB/ SYSTAB	○ A new column, dbspace_id has been added as an eventual replacement to the current file_id column. ○ The file_id column is deprecated. Use dbspace_id instead. For global temporary tables, SYSTAB.file_id now points to the temporary dbspace, instead of the system dbspace. ○ A new column, last_modified_tsn, has been added to store a sequence number for the transaction that modified the table.	See "SYSTAB system view" [*SQL Anywhere Server - SQL Reference*].
ISYSIDX/ SYSIDX	○ A new column, dbspace_id has been added as an eventual replacement to the current file_id column. ○ The file_id column is deprecated. Use dbspace_id instead.	See "SYSIDX system view" [*SQL Anywhere Server - SQL Reference*].
ISYSFILE	This system table is deprecated. All columns, with the exception of lob_map, are now found in the (new) ISYSDBSPACE system table. The lob_map column is now found in the (new) ISYSDBFILE system table.	See: ○ "SYSDBSPACE system view" [*SQL Anywhere Server - SQL Reference*] ○ "SYSDBFILE system view" [*SQL Anywhere Server - SQL Reference*] ○ "SYSFILE compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
ISYSDB-FILE/ SYSDB-FILE	New table to hold information about dbspaces.	See "SYSDBFILE system view" [*SQL Anywhere Server - SQL Reference*].

Table name and/ or view name	Description of change	Additional information
ISYSDB-SPACE/ SYSDB-SPACE	New table to hold information about dbspaces.	See "SYSDBSPACE system view" [*SQL Anywhere Server - SQL Reference*].
SYSDB-SPACE-PERM/ ISYSDB-SPACE-PERM	New table to hold dbspace permissions.	See "SYSDBSPACEPERM system view" [*SQL Anywhere Server - SQL Reference*].
ISYSOB-JECT/ SY-SOBJECT	The file_id column has been renamed to dbspace_id. Also, the object_type column can contain two new values: 17 (Text configuration), and 18 (Dbspace).	See "SYSOBJECT system view" [*SQL Anywhere Server - SQL Reference*].
SYSIN-DEXES	The indextype field now identifies foreign keys and primary key indexes as Primary Key and Foreign Key, respectively to distinguish them from other indexes.	See "SYSINDEXES consolidated view" [*SQL Anywhere Server - SQL Reference*].
ISYSCA-PABILITY-NAME	This table no longer exists in the catalog. The corresponding SYSCAPABILITYNAME system view is still available, but it generated using server properties.	See "SYSCAPABILITYNAME system view" [*SQL Anywhere Server - SQL Reference*].
ISYSE-VENT-TYPE	This table no longer exists in the catalog. The corresponding SYSEVENTTYPE system view is still available, but it generated using server properties.	See "SYSEVENTTYPE system view" [*SQL Anywhere Server - SQL Reference*].
ISYS-VIEW	New column called mv_last_refreshed_tsn to store a sequence number for the transaction that refreshed the materialized view.	See "SYSVIEW system view" [*SQL Anywhere Server - SQL Reference*].

Table name and/ or view name	Description of change	Additional information
ISYSLOGINMAP/ SYSLOGINMAP	New table to hold information about login policies.	See "SYSLOGINMAP system view" [*SQL Anywhere Server - SQL Reference*].
ISYSLOGINPOLICY/ SYSLOGINPOLICY	New table to hold information about login policies.	See "SYSLOGINPOLICY system view" [*SQL Anywhere Server - SQL Reference*].
ISYSLOGINPOLICYOPTION/ SYSLOGINPOLICYOPTION	New table to hold information about login policies.	See "SYSLOGINPOLICYOPTION system view" [*SQL Anywhere Server - SQL Reference*].
ISYSTEXTCONFIG/ SYSTEXTCONFIG	New table to hold information about text configuration objects.	See "SYSTEXTCONFIG system view" [*SQL Anywhere Server - SQL Reference*].
ISYSTEXTIDX/ SYSTEXTIDX	New table to hold information about text indexes.	See "SYSTEXTIDX system view" [*SQL Anywhere Server - SQL Reference*].
ISYSTEXTIDXTAB/ SYSTEXTIDXTAB	New table to hold information about text indexes.	See "SYSTEXTIDXTAB system view" [*SQL Anywhere Server - SQL Reference*].

- **PHP function name changes** All PHP functions have been renamed to have sasql_ as their prefix, instead of sqlanywhere_. The sqlanywhere_ prefix is still allowed in the name when calling a function, but is deprecated. You should change your application to use the new prefix.

- **INSERT ... ON EXISTING UPDATE statement now fires triggers** Previously, when you executed an INSERT ... ON EXISTING UPDATE statement, triggers did not fire if data was updated. Now, the database server fires statement-level after triggers for the updates.

- **REFRESH MATERIALIZED VIEW statement** You can no longer specify STATEMENT SNAPSHOT and READONLY STATEMENT SNAPSHOT as the isolation level for the refresh since the effect of these options is the same as specifying SNAPSHOT for the isolation level. See "REFRESH MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*].

- **REORGANIZE TABLE statement** Attempting to execute multiple REORGANIZE TABLE statements simultaneously on the same table now results in an error.

- **sa_validate system procedure** The check_type, express, and checksum arguments for sa_validate are now obsolete; specifying them no longer has an effect. Checksum validation is now performed by default. Also, when the sa_validate system procedure is called without specifying any arguments, in addition to validating all tables, materialized views, and indexes, the database server also validates the database itself, including checksums. See "sa_validate system procedure" [*SQL Anywhere Server - SQL Reference*].

- **-gss server option** The -gss server option is now supported on Windows XP and later. In previous releases, this option was not supported on Windows operating systems. See "-gss dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-gx server option no longer supported** Support for the -gx server option has been removed in this release. Specifying the -gx option when starting a SQL Anywhere database server results in an error.

- **LazyClose connection parameter default setting is now AUTO** In previous versions, when an application closed a cursor, a round trip to the database server was required unless the LazyClose connection parameter was set to NO. Now, cursor close requests are queued for many cursors by default, eliminating a round trip and resulting in improved performance. The LazyClose connection parameter now accepts three values: YES, NO, and AUTO (the default). YES was the default setting in previous releases. See "LazyClose (LCLOSE) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Embedded SQL import library changes** The Watcom and Borland versions of the DBLIB import libraries are no longer included. These are *dblibtw.lib* and *dblibtb.lib*, respectively. An import definition file (*install-dir\SDK\Lib\Def\dblib.def* file) is provided as a replacement for these import libraries.

- **Database tools import library changes** The Watcom and Borland versions of the database tools import libraries are no longer included. These are *dbtlstw.lib* and *dbtlstb.lib*, respectively. An import definition file (*install-dir\SDK\Lib\Def\dbtool.def*) is provided as a replacement for these import libraries.

- **DBLIB indicator behavior defined when no rows received** On a fetch or execute where no rows are received from the database server (on an error or the end of the result set), indicator values are now unchanged. See "Indicator variables" [*SQL Anywhere Server - Programming*].

- **ODBC SQLGetConnectAttr** Using the ODBC SQLGetConnectAttr call to get the SQL_ATTR_CONNECTION_DEAD attribute now gets the value SQL_CD_TRUE if the connection has been dropped, even if no requests have been sent to the server since the connection was dropped. Determining if the connection has been dropped is done without making a request to the server, and the dropped connection is detected within a few seconds. The connection can be dropped for several reasons, such as an idle timeout.

In previous releases, SQL_ATTR_CONNECTION_DEAD only got the value SQL_CD_TRUE if the connection was disconnected or if the ODBC driver made a request to the server (for example, by calling SQLExecDirect) after the connection was dropped.

- **Databases cannot be created or started that are named utility_db** The name utility_db is now reserved for the SQL Anywhere Server utility database. If you attempt to create a new database or start an existing database named *utility_db.db*, an error is returned. If you have an existing database named utility_db, you can start it with a different name. See "Using the utility database" [*SQL Anywhere Server - Database Administration*].

- **Computed column dependencies** Previously, to allow an update or insert operation to proceed without error, an application could have used triggers to assign non-NULL values to columns that were declared NOT NULL. This impacted computed columns that were dependent on the column, since it could result in a computed value that did not reflect the intended computation. Now, an attempt to set a NULL value in a NOT NULL column that a computed column depends on, fails with an error message and no triggers are fired. See "Inserting into and updating computed columns" [*SQL Anywhere Server - SQL Usage*].

- **Dbspace names containing a period generate an error** In previous releases, if a dbspace name that was not quoted contained a period, then the part of the dbspace name before the period was silently ignored by the server. The database server now generates an error for these names.

- **SQL Anywhere web server no longer supports SSL version 2.0** When using the SQL Anywhere web server, only SSL version 3.0 and TLS version 1.0 connections are supported. SSL version 2.0 connections are not supported.

- **CREATE SERVICE option DATATYPE default value has changed** The default value of the DATATYPE clause has changed from OFF to ON. If you want the old behavior then you must explicitly include DATATYPE OFF in the CREATE SERVICE definition. See "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*].

- **Some secured features renamed** The following secured features have been renamed for this release:

Deprecated name	New name
xp_read_file	read_file
xp_write_file	write_file
unload_table	write_file
load_table	read_file

  For more information, see "-sf dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Checksum behavior changes** For databases created with version 11 or upgraded to version 11, the database server automatically enables checksums to databases running off of media such as

network drives or removable drives. Checksums remain enabled as long as the database resides on such a device. See "Using checksums to detect corruption" [*SQL Anywhere Server - Database Administration*].

- **HTTP connections do not cause databases to stop automatically** In previous releases, when you configured a database to stop automatically, the database would stop if an HTTP connection disconnected and there were no other connections to the database. Databases now stop automatically only when the last command sequence or TDS connection disconnects.

  If the only connection to a database is an HTTP connection, and the database is configured to stop automatically, when the HTTP connection disconnects, the database does not stop automatically. As well, if a database that is configured to stop automatically has an HTTP connection and a command sequence or TDS connection, when the last command sequence or TDS connection disconnects, the database stops, and any HTTP connections are dropped. See "-ga dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*] and "AutoStop (ASTOP) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Database mirroring behavior change** In previous releases, if the connection parameters specified in the -xp option for the primary or mirror server were invalid, the database server would repeatedly attempt to connect, but the connection would never succeed. In this release, if the connection parameters specified in the -xp option are invalid, and there are multiple databases running on the server, then the mirrored database fails to start and does not attempt to reconnect. If the mirrored database is the only database running on the database server, then the database server does not start.

- **Default refresh behavior for materialized views** Previously, the default refresh behavior for materialized views was WITH EXCLUSIVE MODE. Now, default refresh behavior depends on whether the materialized view is defined as IMMEDIATE REFRESH and whether snapshot isolation level is enabled for the database. See "REFRESH MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*].

- **post_login_procedure database option behavior change** The default setting of the post_login_procedure database option is now the sa_post_login_procedure system procedure. See "post_login_procedure option" [*SQL Anywhere Server - Database Administration*].

- **non_keywords database option** In previous releases, in addition to specifying individual keywords, you could also turn off all keywords since a specified release by using the following special values in the list of keywords:

  ```
  keywords_4_0_d, keywords_4_0_c, keywords_4_0_b, keywords_4_0_a,
  keywords_4_0,
   keywords_5_0_01, keywords_5_0
  ```

  These special values are no longer supported. You can still turn off individual keywords. See "non_keywords option" [*SQL Anywhere Server - Database Administration*].

- **quoted_identifier database option setting respected for remote data access** The local setting of the quoted_identifier option now controls the use of quoted identifiers for Adaptive Server Enterprise and Microsoft SQL Server when you are using remote data access. For example, if you set

the quoted_identifier option to Off locally, then quoted identifiers are turned off for Adaptive Server Enterprise. See :

- "Server class aseodbc" [*SQL Anywhere Server - SQL Usage*]
- "Server class mssodbc" [*SQL Anywhere Server - SQL Usage*]

- **Changes to the scope of the precision and scale database options** In previous releases, you could set the precision and scale database options for individual users or specify that the setting had a temporary scope. However, these settings can affect the recoverability of a database. If the temporary or user-level settings differ from the corresponding PUBLIC settings when executing DDL statements that create or alter tables and domains, you may encounter problems while rebuilding the database. The following behavior now applies for the precision and scale database options:

Database server version	Version 10 or earlier database	Version 11 database	Database upgraded to version 11	Unloading a version 10 or earlier database
11	PUBLIC settings allowed User settings allowed Temporary settings *not* allowed	PUBLIC settings allowed User settings *not* allowed Temporary settings *not* allowed	PUBLIC settings allowed User settings *not* allowed Temporary settings *not* allowed	PUBLIC settings unloaded User settings discarded during unload
10 or earlier	PUBLIC settings allowed User settings allowed Temporary settings allowed	N/A	N/A	PUBLIC settings unloaded User settings unloaded

Version 10 and earlier database servers continue to allow you to set the scale and precision options temporarily, as well as for individual users.

Caution
It is recommended that you do not rely on user-level or temporary settings for the precision and scale database options because of the potential problems you can encounter when rebuilding databases, and because of the unpredictable database server behavior that can occur.

See:

- "precision option" [*SQL Anywhere Server - Database Administration*]
- "scale option" [*SQL Anywhere Server - Database Administration*]

- **OPTION clause behavior change** The OPTION clause for the INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements now returns an error if you specify a database option that the clause does not support. See:

  - "INSERT statement" [*SQL Anywhere Server - SQL Reference*]
  - "UPDATE statement" [*SQL Anywhere Server - SQL Reference*]
  - "DELETE statement" [*SQL Anywhere Server - SQL Reference*]
  - "SELECT statement" [*SQL Anywhere Server - SQL Reference*]
  - "UNION statement" [*SQL Anywhere Server - SQL Reference*]
  - "EXCEPT statement" [*SQL Anywhere Server - SQL Reference*]
  - "INTERSECT statement" [*SQL Anywhere Server - SQL Reference*]

- **Rollback log behavior change for read-only databases** In previous releases, operations on read-only databases involving transactional temporary objects were not treated as transactional: no rollback log information was kept for them. In this release, transactional temporary objects in read-only databases have fully-transactional semantics. They are subject to commits, rollbacks, and rollbacks to savepoints.

- **Itanium 64-bit supported platform changes** In previous versions, a full 64-bit version of the software was available for Windows Server 2003 on Itanium II chips, and a deployment release was available on 64-bit Linux and HP-UX operating systems.

  In this release, only the deployment release for 64-bit HP-UX is available.

- **Unload utility (dbunload) behavior changes** In previous releases, the dbunload -ea, -ek, and -ep options had to be specified with the -an or -ar option to control encryption for the new database. Now, if you unload a database, or any part of it, but do not reload it, the -ea, -ek, and -ep option control the encryption of the table data files that are created. When you use these files to reload a database from Interactive SQL, you must specify the encryption key as a parameter to the READ statement. See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

  As well, in previous releases the version of dbunload used to extract a database did not have to be the same version as the database server running the database. Now, when dbunload is used with a version 10.0.0 or later database, the version of dbunload used must match the version of the database server used to access the database. If an older version of dbunload is used with a newer database server, or vice versa, an error is reported.

- **Extraction utility (dbxtract) behavior change** In previous releases the version of dbxtract used to extract a database did not have to be the same version as the database server running the database. Now, when dbxtract is used with a version 10.0.0 or later database, the version of dbxtract used must match the version of the database server used to access the database. If an older version of dbxtract is used with a newer database server, or vice versa, an error is reported.

- **Changes in locking behavior** In previous releases, an UPDATE or DELETE statement executing at isolation level 0 could block on a row lock for a row that was not affected by the statement. It is now less likely for an UPDATE or DELETE statement to take an intent or exclusive lock on a row that is not affected by the statement. When developing applications, you should use caution when using isolation level 0 or 1 with UPDATE and DELETE statements, and ensure that the behavior is

acceptable for your application. See "Locking during updates" [*SQL Anywhere Server - SQL Usage*] and "Locking during deletes" [*SQL Anywhere Server - SQL Usage*].

- **Changes to property names** The following properties have been renamed in this release:

Old name	New name
CacheHitsEng	CacheHits
CacheReadEng	CacheRead
DiskReadEng	DiskRead
ReadHint	DiskReadHint
ReadHintScatter	DiskReadHintPages
ReadHintScatterLimit	DiskReadHintScatterLimit

  For more information, see "Connection, database, and database server properties" [*SQL Anywhere Server - Database Administration*].

- **Language Selection utility (dblang)** In previous releases, this utility was only installed when the International Resources Development Kit (IRDK) was selected during installation. In this release, all international resources and the Language Selection utility (dblang) are installed all the time.

- **Default dbspace for temporary tables and indexes** Temporary tables can only be created in the TEMPORARY dbspace. If you specify the SYSTEM dbspace in the IN clause of the CREATE TABLE statement, the IN clause is ignored, and the temporary table is created in the temporary dbspace. If you specify a user-defined dbspace in the IN clause of the CREATE TABLE statement, an error is returned. As well, the default_dbspace option is ignored when creating temporary objects.

- **Loading data into temporary tables** When loading data into temporary tables you can no longer load a local temporary table that is ON COMMIT DELETE. In previous releases, you could load data into a local temporary table defined with ON COMMIT DELETE ROWS.

  An autocommit is now performed automatically when you run a LOAD TABLE statement; in previous releases, this did not always occur.

- **Database server options** The -uc and -ui server options are now supported on Mac OS X. Previously they were only supported on Linux. On Linux the -ui server option opens the **Server Startup Options** window, displays the database server messages window, and starts the database server whether or not the X window server starts. On Mac OS X -ui displays database server messages in a new window and starts the database server in shell mode if a usable display isn't available. The -uc server option starts the database server in shell mode. See "-uc dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*] and "-um dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Remote data access no longer works with ODBC drivers that do not support UNICODE calls** Remote data access no longer works with ODBC drivers that do not support UNICODE calls.

As a result, with non-UNICODE ODBC drivers, remote data access does not perform any character set translation on data coming in from the ODBC driver.

- **SYSFILE system view** A row for the temporary file is now included in the SYSFILE compatibility view. See "SYSFILE compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].

SQL Anywhere deprecated and discontinued features

- **FORMAT ASCII clause deprecated for LOAD TABLE, UNLOAD TABLE, INPUT, and OUTPUT statements** The FORMAT ASCII clause for the LOAD TABLE, UNLOAD TABLE, INPUT, and OUTPUT statements has been deprecated and is replaced by FORMAT TEXT. Utilities such as dbunload now generate reload scripts containing FORMAT TEXT rather than FORMAT ASCII.

  For the OUTPUT statement, the FORMAT TEXT clause now writes the data in the same file format as FORMAT ASCII did in previous versions. The output formerly created by FORMAT TEXT is no longer available.

- **Database properties** The following database properties are unsupported:

  - MapPages
  - PreserveSource
  - UniqueIdentifier

- **Server properties** The following server properties have been deprecated in this release:

  - MaxMessage
  - Message
  - MessageTime
  - MessageText
  - MessageWindowSize

- **SPX protocol unsupported** Support for the SPX protocol has been removed in this release. As a result, the following protocol options are discontinued:

  - ExtendedName protocol option [ENAME]
  - RegisterBindery protocol option [REGBIN]
  - SearchBindery protocol option [BINSEARCH]

  The following features of the SQL Anywhere .NET Data Provider are discontinued:

  - SACommLinksOptionsBuilder class: SpxOptionsBuilder property
  - SACommLinksOptionsBuilder class: SpxOptionsString property
  - SASpxOptionsBuilder class

- **dbinit -e option unsupported** The dbinit -e option, used for specifying simple encryption when creating a database, is no longer supported. Use the -ea simple option to specify simple encryption. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*].

- **Discontinued database options** Support for the following database options and their corresponding database properties has been removed in this release.

Options	Behavior in this release
ansi_integer_overflow	An overflow now always results in a `SQLSTATE = 22003 - overflow` error. When unloading, or connecting to, older databases with materialized views, the setting of this option is ignored.
ansi_substring	The behavior of the SUBSTRING function now corresponds to ANSI/ISO SQL/2003 behavior. See "SUBSTRING function [String]" [*SQL Anywhere Server - SQL Reference*].
automatic_timestamp	New columns with the TIMESTAMP data type that do not have an explicit default value defined are never given a default value of the Transact-SQL timestamp.
divide_by_zero_error	Division by zero now results in an error with SQLSTATE 22012. When unloading or connecting to older databases with materialized views, the setting of this option is ignored.
float_as_double	In SQL Anywhere, the FLOAT keyword never behaves like Adaptive Server Enterprise's FLOAT keyword when a precision is not specified. SQL Anywhere does not treat FLOAT values the same as DOUBLE values. For Open Client and jConnect connections, this behavior is different from the default behavior in previous releases. When unloading or connecting to older databases with materialized views, the setting of this option is ignored.
optimistic_wait_for_commit	This option is no longer supported.
query_plan_on_open	A plan is no longer returned when an OPEN is done on a cursor. A more complete description can be obtained using the EXPLAIN statement or the PLAN function. See "EXPLAIN statement [ESQL]" [*SQL Anywhere Server - SQL Reference*] and "PLAN function [Miscellaneous]" [*SQL Anywhere Server - SQL Reference*].
ri_trigger_time	Referential integrity actions are now executed after the UPDATE or DELETE.
truncate_with_auto_commit	A COMMIT is now executed both before and after a TRUNCATE TABLE statement is executed.

Options	Behavior in this release
tsql_hex_constant	Hexadecimal constants are now treated as binary typed constants.
uuid_has_hyphens	UUID strings now contain four hyphens. When unloading or connecting to older databases with materialized views, the setting of this option is ignored.
percent_as_comment	In previous releases, the percent sign (%) could be used as a comment marker depending on the setting of the percent_as_comment database option. Now, SQL Anywhere treats the % sign as a modulo operator. See "MOD function [Numeric]" [*SQL Anywhere Server - SQL Reference*].

- **SQLANYSH10 environment variable unsupported** In previous releases, some of the SQL Anywhere software was installed into a shared directory. This location could be specified by the SQLANYSH10 environment variable. Software is no longer installed into a shared directory as part of the installation process, and the SQLANYSH10 environment variable is no longer used.

  When creating a silent install, you no longer need to set the SHARED_DIR location. See "Using a silent install for deployment" [*SQL Anywhere Server - Programming*].

- **sa_get_server_messages system procedure discontinued** In previous releases, you could use the sa_get_server_messages system procedure to return constants from the database server messages window as a result set. You can now use the sa_server_messages system procedure to obtain this information. See "sa_server_messages system procedure" [*SQL Anywhere Server - SQL Reference*].

- **background_priority option deprecated** The background_priority option has been deprecated. Use the priority option instead. See "priority option" [*SQL Anywhere Server - Database Administration*].

- **encrypt_aes_random_iv option unsupported** Support for the encrypt_aes_random_iv database option has been removed in this release. Now, a random IV (initialization vector) is always used.

- **DLL protocol option unsupported** Support for the DLL protocol option has been removed. Windows database servers and clients use Winsock 2.2. Windows Mobile clients use Winsock 1.1. See "Using the TCP/IP protocol" [*SQL Anywhere Server - Database Administration*].

- **SQL Anywhere Broadcast Repeater utility renamed** In version 10, the command to run the SQL Anywhere Broadcast Repeater utility was dbns10. In this release, it is dbns11. See "Broadcast Repeater utility (dbns12)" [*SQL Anywhere Server - Database Administration*].

- **SQLANY10 and SQLANYSAMP10 environment variables renamed** The SQLANY10 and SQLANYSAMP10 environment variables have been renamed SQLANY11 and SQLANYSAMP11, respectively. See:

  - "SQLANY12 environment variable" [*SQL Anywhere Server - Database Administration*]
  - "SQLANYSAMP12 environment variable" [*SQL Anywhere Server - Database Administration*]

MobiLink

The following sections describe the new features, behavior changes, and deprecated features in MobiLink for version 11.0.0.

MobiLink new features

Following is a list of additions to MobiLink introduced in version 11.0.0.

Consolidated databases

- **DB2 mainframe now supported as a consolidated database** MobiLink has long supported DB2 LUW (Linux, Unix, and Windows) as a consolidated database. Now it also supports DB2 mainframe.
- **MySQL now supported as a consolidated database** MobiLink now supports MySQL as a consolidated database.

  See "MySQL consolidated database" [*MobiLink - Server Administration*].
- **MobiLink System Database (MLSD)** Support has been added for a separate database to hold MobiLink system data (MLSD - MobiLink System Database). This feature must be used with Microsoft DTC (Distributed Transaction Coordinator). See "-cs mlsrv12 option" [*MobiLink - Server Administration*].

New system objects

- **New MobiLink server system tables and schema** Following are changes to the MobiLink system tables:
  - Several new MobiLink system tables have been added:
    - ml_qa_delivery_archive
    - ml_qa_repository_archive
    - ml_qa_repository_props_archive
    - ml_qa_status_history_archive
    - ml_server
    - ml_active_remote_id
    - ml_passthrough
    - ml_passthrough_repair
    - ml_passthrough_script
    - ml_passthrough_status

- Several new MobiLink system procedures have been added. See:

  - "ml_add_passthrough system procedure" [*MobiLink - Server Administration*]
  - "ml_add_passthrough_repair system procedure" [*MobiLink - Server Administration*]
  - "ml_add_passthrough_script system procedure" [*MobiLink - Server Administration*]
  - "ml_delete_passthrough system procedure" [*MobiLink - Server Administration*]
  - "ml_delete_passthrough_repair system procedure" [*MobiLink - Server Administration*]
  - "ml_delete_passthrough_script system procedure" [*MobiLink - Server Administration*]

iAnywhere Solutions Oracle driver

- **Oracle DSN can store an encrypted password** When creating an Oracle ODBC data source in the Windows ODBC Administrator, you can now choose to encrypt the password that is stored in the ODBC data source. See "iAnywhere Solutions 12 - Oracle ODBC driver" [*MobiLink - Server Administration*].

- **Oracle ODBC driver supports Microsoft distributed transactions** The Oracle ODBC driver now supports Microsoft distributed transactions. In the Windows ODBC Administrator, select Enable Microsoft distributed transactions and make sure that the appropriate DLL is installed with the Oracle client. See "iAnywhere Solutions 12 - Oracle ODBC driver" [*MobiLink - Server Administration*].

MobiLink server

- **Relay server** The relay server is a set of web extensions that enable secure, load-balanced communication between mobile devices and MobiLink, Afaria and OneBridge servers communicating through a web server. See "Introduction to the Relay Server" [*Relay Server*].

- **Sybase relay server hosting service** The Sybase relay server hosting service is a farm of relay servers hosted by Sybase. It is intended to ease the development of mobile applications that use MobiLink data synchronization and to simplify the evaluation process for developers, especially where data is sent using public wireless networks. See "Sybase Hosted Relay Service" [*Relay Server*].

- **MobiLink Server Farm** MobiLink servers may now be explicitly grouped into server farms of identical servers.

  Redundant, concurrent synchronizations from the same remote ID are now automatically detected across the entire farm. This removes the need for a load balancer to keep sending the same remote ID to the same MobiLink server.

  You can now configure each MobiLink server identically.

  Failover is automatically supported for the Notifier and the QAnywhere connector. The farm automatically appoints a MobiLink server to run these, and will elect a new server to run them if the first appointee computer fails.

- **64-bit Platforms** MobiLink server is now fully 64-bit on several 64-bit platforms. For a list of the supported platforms 64-bit platforms, see http://www.sybase.com/detail?id=1002288.

- **New member for Java DownloadTableData interface** getLastDownloadTime method to return last download time for a table. See "getLastDownloadTime method" [*MobiLink - Server Administration*].

- **SQL passthrough** The SQL Passthrough feature allows you to download scripts of SQL statements from a consolidated database to a SQL Anywhere or UltraLite client, and have those SQL statements executed on the client at an appropriate time.

- **Info message listening** The Java and .NET APIs now allow users to register to receive notifications whenever an info line prefixed with "I" is printed to the log.

  See:

  - "INFO variable" [*MobiLink - Server Administration*]
  - "addInfoListener method" [*MobiLink - Server Administration*]
  - "removeInfoListener method" [*MobiLink - Server Administration*]
  - "MessageType enumeration" [*MobiLink - Server Administration*]
  - "InfoListener event" [*MobiLink - Server Administration*]

New mlsrv11 features

- **-cs option** MobiLink server system objects such as system tables, procedures, triggers, and views can now be stored in a database other than the consolidated database. The database that stores the MobiLink system objects is called the MobiLink System Database or MLSD.

  Use -cs to specify connection parameters for your MLSD. See "-cs mlsrv12 option" [*MobiLink - Server Administration*].

- **-lsc option** Specifies the local server connect information. This information is passed to other servers in the server farm. See "-lsc mlsrv12 option" [*MobiLink - Server Administration*].

- **-ss option** Enables the MobiLink server to run in a server farm.

  Separately licensed component required
  The -ss option is a feature of the MobiLink high availability option, which requires a separate license. See "Separately licensed components" [*SQL Anywhere 12 - Introduction*].

- **-tc option** Set the count down timer for SQL script execution. See "-tc mlsrv12 option" [*MobiLink - Server Administration*].

- **-tf option** Fail the SQL script execution when the count down timer specified with -tc expires (not for Oracle). See "-tf mlsrv12 option" [*MobiLink - Server Administration*].

New MobiLink scripting features

- **Non-blocking download ACK scripts** See "nonblocking_download_ack connection event" [*MobiLink - Server Administration*] and "publication_nonblocking_download_ack connection event" [*MobiLink - Server Administration*].

MobiLink Redirector enhancements

- **Redirector deprecated** The Redirector is deprecated. It has been replaced by the Relay Server. See "Introduction to the Relay Server" [*Relay Server*].

- **New Relay Server** The relay server is a set of web extensions that enable secure, load-balanced communication between mobile devices and MobiLink, Afaria, and OneBridge servers communicating through a web server. See "Introduction to the Relay Server" [*Relay Server*].

MobiLink clients

SQL Anywhere clients

- **dbmlsync APIs for C++ and .NET** The Dbmlsync API provides a programming interface that allows MobiLink client applications written in C++ or .NET to launch synchronizations and receive feedback about the progress of the synchronizations they request. See "Dbmlsync API" [*MobiLink - Client Administration*].

- **Synchronization profiles** You can now store dbmlsync command lines in your database as synchronization profiles. The -sp dbmlsync option enables you to add options from a synchronization profile to command line synchronization options. See "-sp dbmlsync option" [*MobiLink - Client Administration*].

- **LOAD TABLE now optionally logs loaded rows as inserts** The new clause WITH ROW LOGGING means that SQL Anywhere remote databases can now load data using the LOAD TABLE statement. Previously, LOAD TABLE was not always useful in a synchronization environment because the rows were not logged and so they were ignored by dbmlsync. See "Import data with the LOAD TABLE statement" [*SQL Anywhere Server - SQL Usage*].

- **dbmlsync log scanning optimization** Now optimized for non-overlapping publications.

Security

The following security features have been added in this release:

- **End-to-end encryption** MobiLink synchronization streams and clients now support protocol-level end-to-end encryption. See "End-to-end encryption" [*SQL Anywhere Server - Database Administration*].

- **Key Pair Generator utility (createkey)** This utility creates RSA and ECC key pairs for use with MobiLink end-to-end encryption. See "Key Pair Generator utility (createkey)" [*SQL Anywhere Server - Database Administration*].

Server-initiated synchronization

- **Support for MobiLink server farm** SIS has been enhanced to operate better in a MobiLink server farm environment. You can now run a Notifier on every MobiLink server in the farm and the Notifiers

together ensure that there are no redundant notifications to the same Listener. See "-lsc mlsrv12 option" [*MobiLink - Server Administration*].

Listener enhancements

- **New dblsn options** The following options have been added to the MobiLink Listener for Windows:
  - **-ni** Disable IP tracking.
  - **-pc-** Disable persistent connection.
  - **-nu** Disable default UDP listening.

MobiLink behavior changes and deprecated features

Following is a list of changes to MobiLink introduced in version 11.0.0.

MobiLink server changes

- **Change to default download acknowledgement** The default value for the -nba option is now -nba+, which makes non-blocking download acknowledgement the default. If you use download blocking download acknowledgement in an existing deployment, you must either:
  - Use the -nba- option.
  - Look at your synchronization scripts to use the nonblocking_download_ack and/or publication_nonblocking_download_ack scripts (recommended).
- **certificate and certificate_password protocol options renamed** The TLS and HTTPS certificate and certificate_password protocol options have been renamed to identity and identity_password, respectively. See MobiLink servers: "-x mlsrv12 option" [*MobiLink - Server Administration*].

MobiLink client changes

- **Dbmlsync integration component deprecated** The Dbmlsync integration component is deprecated. It has been replaced by the dbmlsync API.

  See "Dbmlsync API" [*MobiLink - Client Administration*].
- **dbmlsync StreamCompression extended option no longer supported** This option is no longer supported.
- **-lt extended option now defaults to OFF** The LockTables (-lt) extended option for dbmlsync formerly defaulted to ON. It now defaults to OFF, meaning that by default, dbmlsync does not lock synchronization tables. See "LockTables (lt) extended option" [*MobiLink - Client Administration*].

Miscellaneous MobiLink behavior changes

Server-initiated synchronization

- **Sierra Wireless Aircards no longer supported** The SMS listening libraries for Sierra Wireless Aircards are no longer supported.
- **-g option deprecated** The -g Listener option has been replaced by the -ni option.
- **-ga option deprecated** The -ga Listener option has been deprecated. Asynchronous IP tracking is now implicit.
- **-gi default changed** The default polling interval has changed from 10 seconds to 60 seconds.

QAnywhere

The following sections describe the new features, behavior changes, and deprecated features in QAnywhere for version 11.0.0.

QAnywhere new features

Following is a list of additions to QAnywhere introduced in version 11.0.0.

New QAnywhere client APIs

- **.NET APIs** The following .NET APIs have been added:
  - CreateQAManager method
  - CreateQAManager(Hashtable String)
  - CreateQATransactionalManager()
  - CreateQATransactionalManager(Hashtable String)
- **Java QAnywhere API has a new interface class** A new class has been added. See "QAMessageListener2 interface" [*QAnywhere*]s.
- **C# QAnywhere API has new delegates**
  - "ExceptionListener delegate" [*QAnywhere*]
  - "ExceptionListener2 delegate" [*QAnywhere*]
  - "MessageListener2 delegate" [*QAnywhere*]

QAnywhere Agent new features

- **UltraLite as a message store** QAnywhere now supports UltraLite as a message store. See "qauagent utility" [*QAnywhere*].
- **qastop -id option** This option allows a message store ID to be given to qastop to have it stop the QAnywhere Agent that is connected to the database with the given store ID.

- **New qauagent utility** QAnywhere agent for UltraLite. See "qauagent utility" [*QAnywhere*].

Other QAnywhere enhancements

- **UltraLite as a message store** QAnywhere now supports UltraLite as a message store. See "qauagent utility" [*QAnywhere*].

- **Incremental upload and download enhancements** Incremental upload and download can break large messages into smaller message pieces. See "-iu qaagent option" [*QAnywhere*] and "-idl qaagent option" [*QAnywhere*].

- **Specify database type as a configuration property** You can now specify the database type of the Client Message Store as a QAnywhere manager configuration property. See "QAnywhere manager configuration properties" [*QAnywhere*].

- **Messages archived once final state is reached** There is now an archive message store that has the sole purpose of storing messages after they have reached a final state and are waiting to be permanently deleted. See "Archive message store requests" [*QAnywhere*].

- **Server delete rules** Server delete rules now apply to the archive message store.

- **Enhancement to server management requests** Server Management Requests have been extended to include an <actionsResponseId> tag which can optionally be included inside an <actions> tag.

QAnywhere behavior changes and deprecated features

Following is a list of changes to QAnywhere introduced in version 11.0.0.

QAnywhere Agent changes

- **qastop utility** If no options are given to qastop, then all QAnywhere Agents running on the same computer will be stopped.

- **Reserved stack sizes for Windows Mobile** For Windows Mobile, the reserved stack sizes for all threads in *qaagent.exe*, *dblsn.exe*, and *dbmlsync.exe* have been reduced.

Other QAnywhere changes

- **Non-blocking download acknowledgements** Successfully downloaded messages now have their status updated from transmitting to transmitted in the server immediately following the synchronization.

SQL Remote

The following sections describe the new features, behavior changes, and deprecated features in SQL Remote for version 11.0.0.

SQL Remote new features

- **Enhancements to Extraction utility (dbxtract) -ea option** The -ea option for dbxtract now accepts both none and simple as encryption types. Specifying none results in no encryption. Specifying simple results in simple encryption. Also, the default encryption type has changed, depending on whether the -ek, -et, or -ep options are specified with -ea. See "Extraction utility (dbxtract)" [*SQL Remote*].

- **-ap, -er, -et, and -nl options added to Extraction utility (dbxtract)** You can now use the -ap, -er, -et, and -nl options with dbxtract. See "Extraction utility (dbxtract)" [*SQL Remote*].

SQL Remote behavior changes and deprecated features

Following is a list of changes to SQL Remote introduced in version 11.0.0.

- **LOAD TABLE now optionally logs loaded rows as inserts** The new clause WITH ROW LOGGING means that SQL Anywhere remote databases can now load data using the LOAD TABLE statement. Previously, LOAD TABLE was not always useful in a synchronization or replicating environment because the rows were not logged and so they were ignored by dbremote or dbltm. See "Import data with the LOAD TABLE statement" [*SQL Anywhere Server - SQL Usage*].

- **VIM and MAPI message types unsupported** Support for the VIM and MAPI message systems has been removed in this release. When you upgrade a database that uses VIM or MAPI to SQL Anywhere version 11.0.0, you must change the message type to File, FTP, or SMTP. *Dbremote.exe* does not start if the message type is MAPI or VIM.

  The simplest change is to switch to SMTP/POP; changing message types may require a change to your mail server to support SMTP/POP.

  For more information about choosing a SQL Remote message type, see "SQL Remote message systems" [*SQL Remote*].

UltraLite

The following sections describe the new features, behavior changes, and deprecated features in UltraLite for version 11.0.0.

UltraLite new features

Following is a list of additions to UltraLite introduced in version 11.0.0.

Main features

- **Support for synchronization profiles** UltraLite 11.0.0 and 11.0.1 support synchronization profiles. See "ALTER SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*], "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*], and"DROP SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **UltraLite SELECT statement** The default for UltraLite SELECT statements that do not explicitly contain a FOR clause is now FOR READ ONLY. This change allows UltraLite to choose more optimal plans for queries when updates are not permitted. See "SELECT statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **UltraLite SYNCHRONIZE statement** A new statement for synchronizing an UltraLite synchronization profile or specific synchronization options. See "SYNCHRONIZE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **UltraLite CREATE SYNCHRONIZATION PROFILE statement** A new statement for creating an UltraLite synchronization profile. Synchronization profiles define how an UltraLite database synchronizes with the MobiLink server. See "CREATE SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **UltraLite ALTER SYNCHRONIZATION PROFILE statement** A new statement for altering an UltraLite synchronization profile. See "ALTER SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **UltraLite DROP SYNCHRONIZATION PROFILE statement** A new statement for deleting an UltraLite synchronization profile. See "DROP SYNCHRONIZATION PROFILE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **Support for SQL Anywhere passthrough scripts** Please note this functionality is not supported in UltraLite 12.

  UltraLite utilities now include support for SQL Anywhere pass-through scripts. The changes apply to the following utilities:

  - ulcond11
  - ulunload
  - ulload
  - ulinfo
  - ulsync

  See:

  - "UltraLite Database Unload utility (ulunload)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Load XML to Database utility (ulload)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Information utility (ulinfo)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Synchronization utility (ulsync)" [*UltraLite - Database Management and Reference*]

- **UltraLite database validation** You can now use the ulvalid utility or ValidateDatabase API to validate an UltraLite database. The validation tests for certain types of corruption in the database file, and you can use command line parameters to refine your results. See "UltraLite Validate Database utility (ulvalid)" [*UltraLite - Database Management and Reference*] and "Validate an UltraLite database" [*UltraLite - Database Management and Reference*].

  UltraLite.NET now supports the ValidateDatabase function. You can now validate a database or specific tables with or without a connection. See "ULDatabaseManager class" [*UltraLite - .NET Programming*] and "ULConnection class" [*UltraLite - .NET Programming*].

  You can now use the **Validate Database Wizard** in Sybase Central to validate an UltraLite database. The **Validate Database** option is available on the **Tools** menu.

- **Support for events and notifications** UltraLite now supports events and notifications. Notification messages are sent to registered queues or connections when events occur. User events may also be defined and triggered by applications. APIs for events and notifications are provided in each supported language. Additionally, a SQL function is provided to access the API functionality.

- **UltraLite support for isolation levels** Now, by default, connections are isolated from each other. Uncommitted changes by other connections and downloads are not visible until committed.

  You can now set the isolation level to READ_COMMITTED or READ_UNCOMMITTED. See "UltraLite isolation levels" [*UltraLite - Database Management and Reference*] and "UltraLite transaction processing" [*UltraLite - Database Management and Reference*].

  UltraLite.NET now supports the ReadUncommitted isolation level. The default isolation level of a connection in auto-commit mode is ReadCommitted. See "UltraLite transaction processing" [*UltraLite - Database Management and Reference*] and "UltraLite isolation levels" [*UltraLite - Database Management and Reference*].

- **UltraLite ALTER DATABASE SCHEMA FROM FILE statement** You can now use the ALTER DATABASE SCHEMA FROM FILE statement to alter an UltraLite schema. The ALTER DATABASE SCHEMA FROM FILE statement replaces the 9.0.2 schema upgrade feature that was implemented with the now removed UpgradeSchemaFromFile or ApplyFile methods. Use either the ulinit or ulunload utilities to ensure that the DDL statements required are syntactically correct.

  See:

  - "ALTER DATABASE SCHEMA FROM FILE statement [UltraLite]" [*UltraLite - Database Management and Reference*]
  - "Deploying UltraLite schema upgrades" [*UltraLite - Database Management and Reference*]
  - "UltraLite Initialize Database utility (ulinit)" [*UltraLite - Database Management and Reference*]
  - "UltraLite Database Unload utility (ulunload)" [*UltraLite - Database Management and Reference*]

- **Extract Database Wizard behavior changes** You can now exclude tables from the extraction process, and the **Extract Database Wizard** now omits publications with duplicate names from the list of available publications. See "Upgrading databases created with previous versions of UltraLite" on page 331.

- **UltraLite client version and build number added to MobiLink log files** During synchronization, UltraLite clients now add their version and build number to the MobiLink server log. See:
  - "Viewing MobiLink server logs" [*MobiLink - Server Administration*]
  - "Synchronization parameters for UltraLite" [*UltraLite - Database Management and Reference*]
- **UltraLite LOAD TABLE statement** The LOAD TABLE statement can now be executed on desktop computers. See UltraLite LOAD TABLE statement. See "LOAD TABLE statement [UltraLite]" [*UltraLite - Database Management and Reference*].
- **Background synchronization support** You can now begin a synchronization on a separate thread at any point in your application and UltraLite will upload only the rows that were committed at the time the upload began. You can now modify the database during the upload and commit your changes without affecting the upload. Any rows committed while the upload is in progress are ignored by the upload. See "Concurrency in UltraLite" [*UltraLite - Database Management and Reference*].
- **Enhanced GUID identifier support** In previous versions of UltraLite, runtime allowed the input and output of UUID (Universally Unique Identifier) or GUID (Globally Unique Identifier) identifiers as either 16-byte binaries or strings. An endian conversion made the identifiers compatible with GUID structs. In UltraLite 11, GUID structs can be explicitly input and output from the runtime and the endian conversion is not required.
- **ul_stream_error struct** In UltraLite 11, the stream_id, stream_context, and error_string_length fields are removed. In addition, the error_string field has been changed from a user-supplied char * to a static char array.

Platforms and devices

- **Support for native amd64/x64 ESQL and C++ application deployment to 64 bit Windows platforms (64 bit XP and later)** UltraLite now supports deploying native amd64/x64 ESQL and C ++ applications to 64-bit Windows platforms (64-bit XP and later). Note, however, that to develop UltraLite applications on a 64-bit machine, you must use the 32-bit versions of the UltraLite utilities. In addition, you will need to use the 32-bit version of DBISQL and Sybase Central if you are connecting to UltraLite on a 64-bit machine.
- **UltraLite utilities ported to Linux (32 bit)** See "Utilities" on page 147.

Security

- **End-to-end encryption** UltraLite now supports protocol-level end-to-end encryption. Data is encrypted using 128-bit AES in cipher block chaining (CBC) mode with key exchange handled via RSA or ECC.

Utilities

- **UltraLite Unload Database to XML utility (ulunload)** You can now use the -s option to unload the schema and save data in a SQL Anywhere-compatible format. See "UltraLite Database Unload utility (ulunload)" [*UltraLite - Database Management and Reference*].

- **UltraLite Initialize Database utility (ulinit)** You can now use the -d option to copy data from a SQL Anywhere database into a new UltraLite database. See "UltraLite Initialize Database utility (ulinit)" [*UltraLite - Database Management and Reference*].

- **ulerase** New utility program to erase an UltraLite database (including any temporary files related to the database). This utility requires a user ID and password to confirm access to the database. See "UltraLite Erase database (ulerase)" [*UltraLite - Database Management and Reference*].

- **ulvalid** New utility program to validate an UltraLite database. Validation tests for certain types of corruption in the database file, and is configurable according to command line parameters. See "UltraLite Validate Database utility (ulvalid)" [*UltraLite - Database Management and Reference*].

- **The following UltraLite utilities have been ported to Linux (and are available in 32-bit versions only)**

  - **ulcreate** create a new, empty UltraLite database
  - **ulerase** permanently erase an UltraLite database and associated checkpoint file
  - **ulinfo** display information about an existing UltraLite database
  - **ulinit** create a new UltraLite database from the schema available in a SQL Anywhere reference database
  - **ulload** create and load an UltraLite database from the XML data saved by ulunload
  - **ulsync** synchronize an UltraLite database with a consolidated database, using MobiLink as the transfer agent
  - **ulunload** unload an UltraLite database to XML
  - **ulvalid** run validity checks on an UltraLite database

The ulunloadold utility is not available on Linux.

SQL

- **IF and CASE statements, and CASE expression enhancement** For improved compatibility, IF expressions are now permitted to end with either ENDIF or END IF. CASE expressions are now permitted to end with either END or END CASE. See "IF expressions" [*UltraLite - Database Management and Reference*] and "CASE expressions" [*UltraLite - Database Management and Reference*].

Programming interfaces

General improvements

Publication masks have been replaced by publication lists. The keyword Publications takes a comma-separated list of publication names.

UltraLite C/C++

A new method, UltraLite_Table* OpenTableEx(), is now part of the UltraLite_Connection object. This method gives non-SQL applications a more versatile way of opening a table and directly scan rows. See "Using direct page scans" [*UltraLite - Database Management and Reference*].

With this method you can specify one of the following ways to open a table:

- To return the rows in of the primary key, use ul_table_open_primary_key.
- To return the rows in arbitrary order, use ul_table_open_no_index.
- To return rows in the order specified by an index, use ul_table_open_with_index.

UltraLite embedded SQL

- Documentation for two functions related to error interpretation. See "ULGetErrorParameter method" [*UltraLite - C and C++ Programming*] and "ULGetErrorParameterCount method" [*UltraLite - C and C++ Programming*].
- Functions ULGetLastDownloadTime, ULResetLastDownloadTime, and ULCountUploadRows have changed syntax to reflect the change from publication masks to publication lists.
- The function ULGetPublicationMask is no longer available.

UltraLite.NET

- ULDataReader class includes a new method: GetRowCount(threshold) to retrieve row count up to a specified maximum number of rows.
- The ULDataReader class now implements the IListSource interface.

UltraLite for M-Business Anywhere

- New methods in Connection class for event handling and notification: cancelGetNotification, createNotificationQueue, declareEvent, destroyNotificationQueue, getNotification, getNotificationParameter, registerForEvent, sendNotification, and triggerEvent.

UltraLiteJ

UltraLiteJ is a Java implementation of UltraLite that supports Java SE and Java ME environments, including BlackBerry smartphones. See "Introduction to UltraLiteJ" [*UltraLiteJ*].

UltraLite behavior changes and deprecated features

Following is a list of changes to UltraLite introduced in version 11.0.0.

Deprecated platforms

- The UltraLite C++ interface no longer supports the Symbian OS. Developers of UltraLite applications for Symbian should use UltraLiteJ. See "Introduction to UltraLiteJ" [*UltraLiteJ*].
- The UltraLite.NET interface no longer supports the .NET 1.0 component. The .NET 1.0 API reference has been removed from the documentation.

UltraLite for AppForge is not supported in version 11.

Database properties

The following database properties have been deprecated in this release:

- CollationName

Connection parameters

The following UltraLite connection parameter has been deprecated in this release.

- ORDERED_TABLE_SCAN

Removed utilities

The **Migrate C++ Applications Wizard** is no longer available in Sybase Central.

Removed, deprecated, and modified functions

- The ULGetPublicationMask function in the ESQL interface is removed (publication masks have been replaced by publication lists).
- Functions ULGetLastDownloadTime, ULResetLastDownloadTime and ULCountUploadRows in the ESQL interface have changed syntax to reflect the change from publication masks to publication lists.
- In the M-Business Anywhere API, the following are deprecated: DatabaseSchema.getTableCountInPublications, DatabaseSchema.SYNC_ALL_DB, DatabaseSchema.SYNC_ALL_PUBS, PublicationSchema.getMask, SyncParms.getPublicationMask, and SyncParms.setPublicationMask.
- In the M-Business Anywhere and .NET APIs, the following methods used to accept a publication mask as a parameter. That parameter has been changed to a publication list (string). The affected methods are in the Connection class: countUploadRows, getLastDownloadTime, and resetLastDownloadTime.
- The ul_sync_info structure has changed: the fields **disable_concurrency**, **checkpoint_store** and **table_order** have been removed. These options are now specified in a new field named **additional_parms** that contains semicolon delimited keyword-value pairs.

Miscellaneous

- **Running a second instance of CustDB** In previous versions, running a second instance of a CustDB application, would cause an error. Now, if you start a second instance of CustDB, the first instance is now brought to the foreground and the second instance exits.

Sybase Central and Interactive SQL

The following sections describe the new features, behavior changes, and deprecated features in Sybase Central and Interactive SQL for version 11.0.0.

Sybase Central and Interactive SQL new features

Following is a list of additions to Sybase Central and Interactive SQL introduced in version 11.0.0.

- **New fast launcher strategy** Previously, the Interactive SQL and Sybase Central fast launchers were started when a user logged in. Now, fast launchers are started only when Interactive SQL or Sybase Central is started. Then, the fast launcher continues to run for 30 minutes (configurable) after the application is shut down. This new strategy speeds up subsequent launches of Interactive SQL and Sybase Central done within the 30 minute window. See "Using the fast launcher option" [*SQL Anywhere Server - Database Administration*].

- **Connect window enhancements and changes** The following enhancements have been made to the **Connect** window in Sybase Central and Interactive SQL:

  - **New Save As ODBC Data Source tool** The **Save As ODBC Data Source** tool allows generate an ODBC data source using the connection parameters that you specify in the **Connect** window. To use this tool, click the **Tools** button on the **Connect** window, select **Save As ODBC Data Source**, and follow the onscreen instructions. See "Create ODBC data sources using the Connect window" [*SQL Anywhere Server - Database Administration*].

  - **New Copy Connection String To Clipboard tool** The **Copy Connection String To Clipboard** tool allows you to copy the connection string to the clipboard to see what is being passed to the database server. To use this tool, click the **Tools** button on the **Connect** window, and select **Copy Connection String To Clipboard**. Paste the connection string in a text editor to view it.

  - **Connect Assistant** There is now a **Connect Assistant** tool in the right half of the **Connect** window. The **Connect Assistant** is a wizard-like feature designed to help you connect to a database.

  - **Enhancements to the Advanced tab** The **Advanced** tab now displays a table of properties, as well as brief descriptions of the properties themselves.

  - **New Networking tab** The **Networking** tab allows you to specify options for the supported protocols: shared memory and TCP/IP. This new tab replaces the functionality offered by the **Search Network For Database Servers** option that was previously located on the **Database** tab.

- **Keyboard shortcuts** Now in the Sybase Central Code Editor and in the **SQL Statements** pane in Interactive SQL, you can use keyboard shortcuts to increase and decrease the indentation of your code and comments. In addition you can use keyboard shortcuts to add and remove both the double hyphen and double slash comment indicators. See "Sybase Central keyboard shortcuts" [*SQL Anywhere Server - Database Administration*].

Sybase Central new features

Following is a list of additions to Sybase Central plug-ins introduced in version 11.0.0.

SQL Anywhere plug-in new features

- **Database Overview tab** You can now obtain a high-level view of the health and statistics of the database server and its features. See "Monitoring database health and statistics" [*SQL Anywhere Server - Database Administration*].

- **Database documentation** You can now generate documentation for your database using the **Database Documentation Wizard**. See "Documenting a database" [*SQL Anywhere Server - Database Administration*].

- **New wizards** The SQL Anywhere plug-in now contains the following new wizards:

  - **Create Login Policy Wizard**
  - **Database Documentation Wizard**
  - **Create Text Index Wizard**
  - **Create Text Configuration Object Wizard**
  - **Create Database Wizard**

- **New properties windows** The SQL Anywhere plug-in now contains properties windows for the following features:

  - Text indexes
  - Text object configurations
  - External environments
  - Materialized views
  - Database auditing

- **Set auditing preferences within Sybase Central** You can now set preferences for database auditing and view auditing information using Sybase Central. See "Controlling auditing" [*SQL Anywhere Server - Database Administration*] and "Retrieving auditing information" [*SQL Anywhere Server - Database Administration*].

- **Support for specifying collation tailoring settings** The **Settings** tab on the **Database Properties** window now allows you to specify collation tailoring options. You can also specify collation tailoring options when creating a new database using the **Create Database Wizard**.

UltraLite plug-in new features

- **New wizards** The UltraLite plug-in now contains the following new wizards:
  - **Validate Database Wizard**
  - **Create Synchronization Profile Wizard**
- **New properties windows** The UltraLite plug-in now contains properties windows for the following features:
  - Synchronization profiles

MobiLink plug-in new features

- **Support for synchronization profiles** You can create and manage synchronization profiles in Sybase Central.
- **New wizards** The MobiLink plug-in now contains the following wizards:
  - **Create Passthrough Script Wizard**
  - **Create Passthrough Download Wizard**
- **New properties windows** The MobiLink plug-in now contains property windows for the following features:
  - Passthrough scripts
  - Passthrough downloads

QAnywhere plug-in new features

- **Support for UltraLite** The QAnywhere plug-in now supports UltraLite as a client message store and supports the new message archive feature. See "Setting up the client message store" [*QAnywhere*].

Interactive SQL new features

Following is a list of additions to Interactive SQL introduced in version 11.0.0.

- **Automatically release database locks** The database server creates schema locks on tables that you view in Interactive SQL, even if you do not modify the table.

  However, now Interactive SQL attempts to release the database schema locks it creates when it displays your result set.

  After you execute a statement that returns a result set, Interactive SQL checks if your connection has any uncommitted changes in the database. If none exist, then Interactive SQL releases your schema locks; otherwise, Interactive SQL does not release your schema locks. That is, Interactive SQL does not release your schema locks if you have any uncommitted changes to the database.

- **Support for read-only result sets** You can now make result sets read-only in Interactive SQL. To do this, choose **Options** » **Results**, and then select the **Disable Editing** option. This setting applies

to subsequently fetched results. See "Editing table values from the Interactive SQL result set" [*SQL Anywhere Server - Database Administration*].

When deploying Interactive SQL, you can prevent users from changing this setting by adding disableResultSetEditing to the entry for lockedPreferences in the *OEM.ini* file. See "Configuring the administration tools" [*SQL Anywhere Server - Programming*].

- **Execute SQL statements one statement at a time** Previously, if you wanted to execute SQL statements one at a time, you repeatedly selected the SQL statement and ran **Execute Selection**. Now you can run **Single Step** to execute the selected statement and to select the next statement for execution. Similar to **Execute Selection**, **Single Step** is available from the **SQL** menu in Interactive SQL. See "Executing SQL statements from Interactive SQL" [*SQL Anywhere Server - Database Administration*].

- **SQL file and database connection favorites** You can now create and maintain a list of favorite database connections and a list of favorite SQL command files with the **Favorites** menu in Interactive SQL. See "Using favorites" [*SQL Anywhere Server - Database Administration*].

- **Disable table editing of result sets** When deploying Interactive SQL, you can now disable table editing of SQL Anywhere and UltraLite result sets in Interactive SQL. See "Configuring the administration tools" [*SQL Anywhere Server - Programming*].

- **New keyboard shortcuts** New keyboard shortcuts have been added to Interactive SQL. See "Interactive SQL keyboard shortcuts" [*SQL Anywhere Server - Database Administration*].

- **Support for preventing option changes in client applications** You can now prevent users from changing some of the Interactive SQL option settings by locking the settings in the *OEM.ini* file. See "Configuring the administration tools" [*SQL Anywhere Server - Programming*].

- **New -version option for dbisql** From the command prompt, type `dbisql -version` to see the version number of Interactive SQL. See "Interactive SQL utility (dbisql)" [*SQL Anywhere Server - Database Administration*] and "Interactive SQL for UltraLite utility (dbisql)" [*UltraLite - Database Management and Reference*].

Sybase Central and Interactive SQL behavior changes and deprecated features

Following is a list of changes to Sybase Central and Interactive SQL introduced in version 11.0.0.

- **Database tool launcher executables are easier to redeploy** The launcher executables for the Sybase Central, Interactive SQL, the Database Console utility, and the MobiLink Monitor are now easier to redeploy. Registry entries and a set directory structure for the location of the JAR files are no longer required. Each executable needs to have a corresponding *filename.INI* file in the same directory (with the same name) as the *filename.exe* file. The *.INI* file contains the details on how to load the tool. See "Deploying administration tools" [*SQL Anywhere Server - Programming*].

- ***OEM.ini* [help] section no longer supported** The [help] section in the *OEM.ini* file is no longer supported. For more information, see "Configuring the administration tools" [*SQL Anywhere Server - Programming*].

Sybase Central behavior changes and deprecated features

Following is a list of changes to Sybase Central introduced in version 11.0.0.

- **Sybase Central configuration file renamed** The *.screpository* file is now named *.screpository600*.

SQL Anywhere plug-in changed features

- **Enhancements to properties windows** The following property pages have been updated:
  - **Web Service Properties** window
  - **User Properties** window
  - **View Properties** windows
  - **Materialized View Properties** window
- **Debugging specific users** When you enter **Debug** mode in the SQL Anywhere plug-in, you must specify what users you want to debug. See "Tutorial: Getting started with the debugger" [*SQL Anywhere Server - SQL Usage*].
- **Updated wizards** The following wizards have been updated:
  - **Create User Wizard**
  - **Create Web Services Wizard**
  - **Deployment Wizard**

MobiLink plug-in changed features

- **Updated wizards** The following wizards have been updated:
  - **Deploy Synchronization Model Wizard**

SQL Anywhere plug-in deprecated features

- **Removed properties tabs** The following property windows have been updated:
  - **Table Properties** window
  - **UltraLite Project Properties** window
  - **UltraLite Statement Properties**window

Interactive SQL behavior changes and deprecated features

Following is a list of changes to Interactive SQL introduced in version 11.0.0.

- **Graphical plans are now viewable in a Plan Viewer** You now view graphical plans for SQL Anywhere databases in a separate, resizable window called the Plan Viewer, in Interactive SQL. This

change makes it easier to view and compare plans because you can now open multiple Plan Viewer windows at the same time. To access the Plan Viewer, select **Tools** » **Plan Viewer**. Text plans for UltraLite databases are also displayed in the Plan Viewer. See "Using the Plan Viewer to view graphical plans" [*SQL Anywhere Server - Database Administration*].

In addition the Interactive SQL option isql_plan option is unsupported.

- **Support for viewing long and short plans has been removed** You can no longer view text plans for SQL Anywhere databases in Interactive SQL. However, you can still retrieve them using the EXPLANATION and PLAN functions. You can still view text plans for UltraLite databases using the Plan Viewer in Interactive SQL.

- **Printing execution plans and result sets** Now you can print the contents of the **SQL Statements** pane and the result sets by pressing Ctrl+P or by choosing **Print** from the **File** menu. Previously you could only print the contents of the **SQL Statements** pane. You can print in the Plan Viewer by pressing the **Print** button. See "Printing SQL statements, execution plans, and result sets" [*SQL Anywhere Server - Database Administration*].

- **Line numbers added to the SQL Statements pane** Line numbers are now visible on the left side of the **SQL Statements** pane. These line numbers can help you identify the location of syntax errors.

- **Enhancement to the Execute SQL Statements toolbar button** Previously, on the Interactive SQL toolbar, the **Execute SQL Statements** button executed all SQL statements. Now you can specify whether to execute all statements, or to execute only the selected statements when the button is clicked.

  To set the behavior of the **Execute SQL Statements** button, from the **Tools** menu, choose **Options** » **Toolbar**. See "Executing SQL statements from Interactive SQL" [*SQL Anywhere Server - Database Administration*].

- **Enhancement to executing batch statements**

  - Interactive SQL provides improved feedback when executing batches of statements. When executing SQL statements from the **SQL Statements** pane, the statement being executed is now selected and scrolled into view. When executing script files via **File** » **Run Script**, a status window appears which shows the progress through the script. See "Executing multiple SQL statements" [*SQL Anywhere Server - Database Administration*].

- **Enhancements to the Results pane**

  - In the **Results** pane, you can now select all the results by pressing Ctrl+A. You can also select the entire result set, not just the currently fetched results. When the **Results** pane does not contain the entire result set, you are prompted to fetch the remaining results. Otherwise, only the currently fetched results are selected.

  - Now when you copy cells from the **Results** tab the copied data is formatted based on the following Interactive SQL options: isql_field_separator, isql_quote, and isql_escape_character. You can also copy to the clipboard selected values, rows, and columns from the result set. See "Copying columns, rows, and cells from an Interactive SQL result set" [*SQL Anywhere Server - Database Administration*].

- Now when you click a column-header in the **Results** tab, the results are sorted by that column. When the **Results** pane does not contain the entire result set, you are prompted to fetch the remaining results. Otherwise, only the currently fetched results are sorted.

- Now you can generate and copy to the clipboard INSERT, DELETE, and UPDATE statements that are based on selected rows in the result set. See "Generating SQL statements from result sets" [*SQL Anywhere Server - Database Administration*].

- The **Results** pane in Interactive SQL has been enhanced to include the following features, available from the right-click menu:

  - **Copy » Copy Cell** Copies the contents of the selected cell.

  - **Copy » Copy Column** Copies cell values from the column the selected cell.

  - **Generate » INSERT Statement** Generates an INSERT statement for each selected row and copies them to the clipboard.

  - **Generate » DELETE Statement** Generates a DELETE statement for each selected row and copies them to the clipboard.

  - **Generate » UPDATE Statement** Generates an UPDATE statement for each selected row and copies them to the clipboard. The generated statements set the column values to their current values. Consequently, executing the statements would not actually change the column values. This functionality can be useful for providing a template UPDATE statement which you could edit before executing it.

  See "Copying columns, rows, and cells from an Interactive SQL result set" [*SQL Anywhere Server - Database Administration*] and "Generating SQL statements from result sets" [*SQL Anywhere Server - Database Administration*].

- **Enhancements to Interactive SQL statements**

  - **DESCRIBE statement enhancements** The DESCRIBE statement can now return information about the database or database server that is connected to Interactive SQL. See "DESCRIBE statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

  - **INPUT and READ statement enhancements** The INPUT and READ statements now attempt to resolve relative paths in two ways. See "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*] and "READ statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

  - **Enhancements to the INPUT and OUTPUT statements**

    - **New support for importing from, and exporting to, ODBC sources** You can now specify an ODBC data source when importing into, and exporting from, the database using the INPUT and OUTPUT statements. To do so, use the new USING clause. See "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*] and "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

You can also specify an ODBC data source when using the **Import Wizard** and the **Export Wizard**. See "Import data with the Import Wizard" [*SQL Anywhere Server - SQL Usage*] and "Export data with the Export Wizard" [*SQL Anywhere Server - SQL Usage*].

- **New support for byte order mark (BOM)** You can now control whether a byte order mark (BOM) in data is processed. To do so, use the new BYTE ORDER MARK clause. See "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*] and "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

- **Supported formats for the INPUT statement have changed** The INPUT statement no longer supports the dBase, Lotus, Excel, and FoxPro file formats. TEXT and FIXED are still supported. If you want to continue to use these file formats, you will have to do so via an ODBC driver. See "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

- **Supported formats for the OUTPUT statement have changed** The OUTPUT statement no longer supports the dBase, Lotus, Excel, and FoxPro file formats. TEXT, FIXED, HTML, SQL, and XML are still supported. See "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

- **ASCII format is renamed TEXT for INPUT and OUTPUT statements** INPUT and OUTPUT statements now use TEXT. Use of ASCII is still supported; however, for backward compatibility.

- **Changes to the Import Wizard and the Export Wizard** When the **Import Wizard** or the **Export Wizard** finishes, the SQL statement generated by the wizard is stored in the command history. To view the generated SQL Statement, choose **SQL** » **History**.

- **Interactive SQL options**

  - **isql_allow_read_client_file and isql_allow_write_client_file** These two options describe how Interactive SQL responds to requests to read and write client-side files. See "isql_allow_read_client_file option [Interactive SQL]" [*SQL Anywhere Server - Database Administration*] and "isql_allow_write_client_file option [Interactive SQL]" [*SQL Anywhere Server - Database Administration*].

  - **-codepage option deprecated** If you want Interactive SQL to read a file with a specific code page, use the ENCODING clause of the INPUT, OUTPUT, or READ statement. See:

    - "Interactive SQL utility (dbisql)" [*SQL Anywhere Server - Database Administration*]
    - "Interactive SQL for UltraLite utility (dbisql)" [*UltraLite - Database Management and Reference*]
    - "INPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*]
    - "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*]
    - "READ statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*]

  - **isql_plan option unsupported** The Interactive SQL option isql_plan option is no longer supported. Attempts to set it are silently ignored for backward compatibility. See "Using the Plan Viewer to view graphical plans" [*SQL Anywhere Server - Database Administration*].

- **SET OPTION statement PUBLIC keyword removed** Support for the PUBLIC keyword is removed for setting Interactive SQL options using the SET OPTION statement. See "Interactive SQL options" [*SQL Anywhere Server - Database Administration*].

- **Changes to the Interactive SQL launcher** The executable for the Windows version of the Interactive SQL launcher has changed from *dbisqlg.exe* to *dbisql.exe*.

  The executable for the command-line version of the Interactive SQL launcher changed from *dbisql.exe* to *dbsql.com*. Batch scripts should call *dbisql* or *dbisql.com*, not *dbisql.exe*.

SQL Anywhere Console utility behavior changes

Following is a list of changes the SQL Anywhere Console utility introduced in version 11.0.0.

- **New Console options** You can now specify a date and time for the database server to shut down from the **Options** window. From the **File** menu, choose **Options** » **Console**.

MobiLink Monitor behavior changes

Following is a list of changes to the MobiLink Monitor introduced in version 11.0.0.

The MobiLink Monitor cannot read monitor files that were created with a version 9 or earlier MobiLink server; the MobiLink Monitor should only be used with MobiLink servers of the same major version. In addition, the worker column has been removed and the following MobiLink Monitor properties have been renamed:

Old property name	New property name
preload_upload	sync_request
verify_upload	authenticate_user

Documentation enhancements

- **Documentation directory enhancements** Previously the documentation existed in *install-dir* *docs*; now the documentation resides in *install-dir**documentation*. The file names for the individual HTML Help and PDF files are also updated.

The new file names for the HTML Help are the following:

- *dbadmin_en11.chm*
- *dbprogramming_en11.chm*
- *dbreference_en11.chm*
- *dbusage_en11.chm*
- *mlclient_en11.chm*
- *mlserver_en11.chm*
- *mlsisync_en11.chm*
- *mlstart_en11.chm*
- *qanywhere_en11.chm*
- *sachanges_en11.chm*
- *saerrors_en11.chm*
- *saintro_en11.chm*
- *scplugin_en11.chm*
- *sqlanywhere_en11.chm*
- *sqlremote_en11.chm*
- *uladmin_en11.chm*
- *ulc_en11.chm*
- *uldotnet_en11.chm*
- *ulj_en11.chm*
- *ulmbus_en11.chm*

- **Supported platform pages are now accessed through the web site** Previously, the supported platform pages were installed with the software. Now all supported platform information is available on the Sybase web site: http://www.sybase.com/detail?id=1002288.

- **Locate button** Now in the HTML Help browser, you can use the **Locate** button to see where the current help page is located in the Table of Contents.

Product-wide features

The following sections describe the new features, behavior changes, and deprecated features that affect all components of SQL Anywhere version 11.0.0.

Product-wide new features

Following is a list of product-wide additions introduced in version 11.0.0.

- **Error reporting enhancements** On Windows, Windows Mobile, and Linux, when an error report is generated, a window appears where you can view the contents of the error report before choosing whether to submit it to iAnywhere. See "Error reporting in SQL Anywhere" [*SQL Anywhere Server - Database Administration*].

You can now configure the Support utility (dbsupport) using the -ce option, which sends an email when dbsupport is monitoring an application and that application crashes. See "Support utility (dbsupport)" [*SQL Anywhere Server - Database Administration*].

- **Additional Windows Mobile platform support** SQL Anywhere now supports Windows Mobile 5 for smartphone and Windows Mobile 6 Standard edition. For information about running SQL Anywhere Server on Windows Mobile, see "SQL Anywhere for Windows Mobile" [*SQL Anywhere Server - Database Administration*] and "Installation considerations: Limitations on Windows Mobile 5.0 for smartphone" [*SQL Anywhere Server - Database Administration*].

- **New table in the SQL Anywhere sample database (*demo.db*)** A new table, MarketingInformation, has been added to the SQL Anywhere sample database. Each row in this table holds an HTML page describing a product in the Products table. This table was added to provide richer character data to query on when testing and trying out features. See "SQL Anywhere sample database" [*SQL Anywhere 12 - Introduction*].

Product-wide behavior changes

Following is a list of product-wide changes in version 11.0.0.

- **Windows CE has changed to Windows Mobile** The name Windows CE has been changed to be Windows Mobile in the documentation and the software, except where it is more accurate to continue referring to Windows CE.

- **Readcert, gencert, and reqtool removed** The utilities readcert, gencert, and reqtool have been removed. They were previously deprecated. In their place, you can use createcert and viewcert. See "Certificate utilities" [*SQL Anywhere Server - Database Administration*].

- **Createcert and viewcert utilities supported on Mac OS X** The createcert and viewcert certificate utilities are now supported on Mac OS X. See "Certificate utilities" [*SQL Anywhere Server - Database Administration*].

- **certificate and certificate_password protocol options renamed** The TLS and HTTPS certificate and certificate_password protocol options have been renamed to identity and identity_password, respectively. See:

  - SQL Anywhere database servers: "-ec dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]
  - SQL Anywhere web servers: "-xs dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]
  - SQL Anywhere protocol options: "Identity protocol option" [*SQL Anywhere Server - Database Administration*] and "Identity_Password protocol option" [*SQL Anywhere Server - Database Administration*]
  - MobiLink servers: "-x mlsrv12 option" [*MobiLink - Server Administration*]

- **Sample identity file changes** The identity files containing the sample certificates and corresponding private key for TLS have been renamed in this release. The file *rsaserver.crt* has been

renamed *rsaserver.id*, and the file *sample.crt* has been renamed *eccserver.id*. The password for both of these identity files has been changed from **tJ1#m6+W** to **test**.

- **Changes to installation directories** 32-bit software is now installed to the *bin32* directory instead of the *win32* directory, and 64-bit software is installed to the *bin64* directory, instead of the *X64* directory. For example, in previous versions, software that was installed to *C:\Program Files \SQL Anywhere 11\win32* is now installed to *C:\Program Files\SQL Anywhere 11\bin32*.

- **Changes to ODBC data sources for the sample databases** In previous releases, the ODBC data sources for the sample databases that are installed with the software were user data sources. The SQL Anywhere 11 Demo, SQL Anywhere 11 CustDB, and QAnywhere 11 Demo data sources are now system data sources.

- **.NET 1.0 unsupported** SQL Anywhere 11 does not support Microsoft Visual Studio .NET 2002 or Visual Studio .NET 2003. However, Microsoft Visual Studio 2005 (.NET 2.0) and Visual Studio 2008 (.NET 3.x) are supported.

What's new in version 10.0.1

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

SQL Anywhere

MobiLink

QAnywhere

UltraLite

SQL Remote

Product-wide features

SQL Anywhere

The following sections describe the new features, behavior changes, and deprecated features in SQL Anywhere for version 10.0.1.

New features

Following is a list of additions to SQL Anywhere databases and database servers introduced in version 10.0.1.

Encryption enhancements

The following changes have been made to enhance support for encryption.

- **Extension to the CREATE DATABASE statement ENCRYPTION clause** The syntax for the ENCRYPTION clause of the CREATE DATABASE statement has been extended to allow you to specify SIMPLE as an encryption type. Additionally, you can specify the encryption key and the algorithm in any order. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to dbinit and dbunload -ea option** The -ea option for dbinit and dbunload now accepts both none and simple as encryption types. Specifying none results in no encryption. Specifying simple results in simple encryption. Also, the default encryption type has changed, depending on whether the -ek, -et, or -ep options are specified with -ea. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*], and "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

  The -e option is deprecated. See "Behavior changes and deprecated features" on page 169.

- **Strong encryption on Mac OS X** You can now encrypt client/server communications using RSA encryption on Mac OS X. See "Encrypting SQL Anywhere client/server communications" [*SQL Anywhere Server - Database Administration*].

Support for client statement caching

Client statement caching is now supported and enabled by default, so that when the same SQL text is prepared and dropped repeatedly, the client caches the statement, leaving it prepared on the server after it has been dropped by the application. This saves the database server the extra work of dropping and re-preparing the statement. Both a version 10.0.1 client library and a version 10.0.1 database server are required to use client statement caching.

The following changes have been made to provide support for client statement caching.

- **max_client_statements_cached option** This option specifies the maximum number of statements which can remain cached (prepared) on the database server even though they have been dropped by the application. See "max_client_statements_cached option" [*SQL Anywhere Server - Database Administration*].

- **New connection and server properties** The ClientStmtCacheHits, ClientStmtCacheMisses, and max_client_statements_cached properties have been added. See "Connection properties" [*SQL Anywhere Server - Database Administration*] and "Database server properties" [*SQL Anywhere Server - Database Administration*].

- **New request statistics** The Statement Cache Hits and Statement Cache Misses statistics have been added. See "Request statistics" [*SQL Anywhere Server - SQL Usage*].

SQL Flagger enhancements

The SQL Flagger feature has been enhanced to allow better detection of compatibility, and to add support for later standards. For example, you can now test compatibility with a specific SQL standard, or compatibility with UltraLite SQL.

To support these enhancements, the following changes have been made:

- **New SQLFLAGGER function** You can use the new SQLFLAGGER function to test that a SQL statement conforms to a specified SQL standard, without actually running the statement. See "SQLFLAGGER function [Miscellaneous]" [*SQL Anywhere Server - SQL Reference*].

- **New sa_ansi_standard_packages system procedure** Using the new sa_ansi_standard_packages system procedure, you can specify a SQL standard and a SQL statement, and obtain the list of non-core SQL extensions that would be used during the execution of the statement. See "sa_ansi_standard_packages system procedure" [*SQL Anywhere Server - SQL Reference*].

  You must upgrade your database to use this feature. See "Upgrading version 10 and later databases" on page 315.

- **New values for the sql_flagger_error_level and sql_flagger_warning_level database options** Several new values are available for the sql_flagger_error_level and sql_flagger_warning_level database options to support the SQL/1999 and SQL/2003 standards. See "sql_flagger_error_level option" [*SQL Anywhere Server - Database Administration*] and "sql_flagger_warning_level option" [*SQL Anywhere Server - Database Administration*].

- **New values for the SQL preprocessor (sqlpp) -e and -w options** Several new values are available for the -e and -w options of the SQL preprocessor (sqlpp) to support the SQL/1999 and SQL/2003 standards. See "SQL preprocessor" [*SQL Anywhere Server - Programming*].

SQL statements

The following enhancements have been made to SQL statements and functions.

- **START DATABASE statement enhancement** The START DATABASE statement now supports a DIRECTORY clause that lets you specify the directory where the database's dbspace files are located. See "START DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements include an OPTION clause** The INSERT, UPDATE, DELETE, SELECT, UNION, EXCEPT, and INTERSECT statements support an OPTION clause that controls how materialized views are used by the statement, specifies how the query is optimized, and can override the settings of the following database options:

  - isolation_level
  - max_query_tasks
  - optimization_goal
  - optimization_level
  - optimization_workload

See:

- "INSERT statement" [*SQL Anywhere Server - SQL Reference*]
- "UPDATE statement" [*SQL Anywhere Server - SQL Reference*]
- "DELETE statement" [*SQL Anywhere Server - SQL Reference*]
- "SELECT statement" [*SQL Anywhere Server - SQL Reference*]
- "UNION statement" [*SQL Anywhere Server - SQL Reference*]
- "EXCEPT statement" [*SQL Anywhere Server - SQL Reference*]
- "INTERSECT statement" [*SQL Anywhere Server - SQL Reference*]

- **HTML_DECODE function** The HTML_DECODE function now decodes more Unicode codepoints given as numeric entities, such as the trademark symbol (™). If a codepoint cannot be represented in the database CHARacter set, it is left in its codepoint form. Previously, codepoints less than 0x7F were converted to CHARacters (for some CHARacter sets, codepoints less than 0xFF were converted to CHARacters), while all other codepoints remained in their codepoint form. See "HTML_DECODE function [Miscellaneous]" [*SQL Anywhere Server - SQL Reference*].

Support for collation tailoring

SQL Anywhere now supports collation tailoring when creating a database. The following changes have been made to support collation tailoring:

- **Enhancements to CREATE DATABASE statement** When creating a database using the CREATE DATABASE statement, or the Initialization utility (dbinit), you can now specify tailoring options for additional control over the sorting and comparing of CHARacters.

  For the CREATE DATABASE statement, collation tailoring is supported using the COLLATION and NCHAR COLLATION clauses. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

  For the Initialization utility, collation tailoring is supported using the -z and -zn options. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*].

  Note
  Databases created with collation tailoring options cannot be started using a pre-10.0.1 database server.

  If you want to use collation tailoring in an existing database, you must create a new version 10.0.1 database that supports collation tailoring, unload the existing database, and then reload the database into the new version 10.0.1 database. See "Rebuilding version 10 and later databases" on page 307.

- **New HasCollationTailoring database property** A new database property, HasCollationTailoring, indicates whether tailoring support was enabled when creating the database. See "Database properties" [*SQL Anywhere Server - Database Administration*].

- **New extended property values** The following new DB_EXTENDED_PROPERTY values are available when querying the Collation, NCHARCollation, and CatalogCollation database properties: CaseSensitivity, AccentSensitive, PunctuationSensitivity, Properties, and Specification. See "DB_EXTENDED_PROPERTY function [System]" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the SORTKEY and COMPARE functions** In addition to accepting a collation name as a parameter, the SORTKEY and COMPARE functions now accept the same parenthesized set of collation tailoring options as the CREATE DATABASE statement. See "SORTKEY function [String]" [*SQL Anywhere Server - SQL Reference*] and "COMPARE function [String]" [*SQL Anywhere Server - SQL Reference*].

Enhancements to web services

The following enhancements have been made to improve the configurability of HTTP and SOAP headers:

- **Improved configurability** The new SET clause of the CREATE PROCEDURE and CREATE FUNCTION statements lets you modify the following options for the HTTP and SOAP protocols: the HTTP version used by the client, whether to use chunking, and, for SOAP requests, the name of the SOAP operation to call, if it is different from the name of the procedure or function. See "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*].

- **HTTP header specification** The syntax for the HEADER clause of the CREATE PROCEDURE and CREATE FUNCTION statements has been extended to allow you to suppress a given HTTP request header, or to provide an empty value for it. This functionality extends to HTTP request headers that are generated automatically, which were not modifiable in previous releases. See "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*] and "HTTP request header management" [*SQL Anywhere Server - Programming*].

- **Support for data types for the SOAP:RPC client** Data typing can be enabled using the DATATYPE clause of the CREATE SERVICE statement. Data type information is included in the XML encoding of parameter input and result set output or responses for all SOAP service formats. This simplifies parameter passing from SOAP toolkits by not requiring client code to explicitly convert parameters to Strings. See "Working with data types (SOAP only)" [*SQL Anywhere Server - Programming*].

- **HTTPS supported on Mac OS X** In previous releases, only the HTTP protocol was supported for Mac OS X. You can now use HTTPS when running the SQL Anywhere database server as a web server on Mac OS X. See "-xs dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*] and "Using SQL Anywhere as an HTTP web server" [*SQL Anywhere Server - Programming*].

Enhancements to database mirroring

The following enhancements have been added to the database mirroring feature:

- **Specifying a preferred database server for database mirroring** You can now specify which database server should assume the role of primary server in the mirroring system. See "-xp dbsrv12 database option" [*SQL Anywhere Server - Database Administration*] and "Specifying a preferred database server" [*SQL Anywhere Server - Database Administration*].

- **Initiating a database mirroring failover from the primary server to the mirror server** You can now initiate a failover from the primary server to the mirror server using the SET PARTNER FAILOVER clause of the ALTER DATABASE statement. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*] and "Initiating failover on the primary server" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere plug-in

- **Triggers folder column names changed** In the Triggers folder, the Table Name and Table Owner columns have been replaced with Object Name, Object Owner, and Object Type columns. The Object Type column does not appear by default, but can be displayed by choosing View » Choose Columns.

- **Triggers tab added to the View Properties window** The properties window for non-materialized views now has a Triggers tab that lists the view's INSTEAD OF triggers.

- **INSTEAD OF trigger support added to Create Trigger wizard** Several enhancements have been made to the Create Trigger wizard to support INSTEAD OF triggers, including the option of choosing whether you are creating a trigger for a table or a non-materialized view. See "Creating triggers" [*SQL Anywhere Server - SQL Usage*].

- **Collation tailoring support added to Create Database wizard** The Create Database wizard now includes a Collation Tailoring page if the selected database server is a version 10.0.1 or later server, or if you have chosen to create the database on the local computer by starting a new database server.

Miscellaneous enhancements

- **Plan caching for simple DML statements** Plan caching has been extended to include INSERT, UPDATE, and DELETE statements that qualify for query bypass (simple statements). See "Plan caching" [*SQL Anywhere Server - SQL Usage*].

- **Materialized views with left and right outer joins now eligible for use during cost-based optimization** Previously, left and right outer joins were allowed in the definition of a materialized view. However, this disqualified the materialized view for use in cost-based optimization. Now, materialized views that have left or right outer joins can be used during cost-based optimization. See "Improving performance with materialized views" [*SQL Anywhere Server - SQL Usage*].

- **Support for INSTEAD OF triggers** A BEFORE or AFTER trigger fires before or after the triggering operation, respectively. An INSTEAD OF trigger replaces the triggering operation. INSTEAD OF triggers can give you more control over the trigger's behavior during insert, update, or delete operations. See "CREATE TRIGGER statement" [*SQL Anywhere Server - SQL Reference*].

- **DBTools enhancement** You can now determine whether a database was created using SQL Anywhere 10.0.0, or a prior version, without starting the database, by using the DBCreatedVersion function. See "DBCreatedVersion function" [*SQL Anywhere Server - Programming*].

- **OLAP enhancements** Two new window aggregate functions, FIRST_VALUE and LAST_VALUE, are now supported. These functions return the first or last value, respectively, of a window, eliminating the need to return these values using self-joins. You can then use these values as baselines in further calculations performed on the window. See "FIRST_VALUE function [Aggregate]" [*SQL Anywhere Server - SQL Reference*] and "LAST_VALUE function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].

- **Enhanced Unix support for IPv6** On Unix you can specify either an interface identifier or interface name as part of the IPv6 address. On Linux (kernel 2.6.13 and later), an interface identifier is

required when you specify an IP address on the client or server (for example, when using the HOST=, MYIP=, or BROADCAST= TCP protocol options). See "IPv6 support in SQL Anywhere" [*SQL Anywhere Server - Database Administration*].

- **Support for TDS DATE and TIME data types** The TDS DATE and TDS TIME data types were recently introduced into TDS clients. Applications that use Open Client 15 or later versions or EBFs of jConnect can now fetch date and time columns as TDS DATE or TDS TIME values instead of TDS DATETIME.

  SQL Anywhere has been enhanced so that TDS-based applications can fetch date and time data as TDS DATE and TDS TIME values. Applications that use older versions of Open Client or jConnect will continue to fetch date and time data as TDS DATETIME. Note that non-TDS-based applications (applications that use embedded SQL, ODBC, or the iAnywhere JDBC driver) have always been able to fetch date and time data as date and time values.

- **New dbinit option to list available CHARacter set encodings** Use the Initialization utility (dbinit) -le option to list the available CHARacter set encodings for a database. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*].

- **New -ds server option** The -ds server option allows you to specify the directory where the dbspace files for a database are located. See "-ds dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*].

- **SADbType.Xml data type** The SADbType.Xml enumeration constant has been added to the SQL Anywhere .NET provider.

- **Units are supported for dynamic traps for the SQL Anywhere SNMP Extension Agent** When setting dynamic traps, you can now use k, m, g, or t to specify units of kilobytes, megabytes, gigabytes, or terabytes when specifying a numeric value for the trap. See "Creating dynamic traps" [*SQL Anywhere Server - Database Administration*].

- **Creating ODBC data sources for the iAnywhere Solutions Oracle driver** You can now use the Data Source utility (dbdsn) to create ODBC data sources for the iAnywhere Solutions Oracle driver by specifying the -or option. See "Data Source utility (dbdsn)" [*SQL Anywhere Server - Database Administration*].

- **Enhancement to window for submitting error reports** The dbsupport window that prompts you to submit an error report to iAnywhere now includes a View Error Report button, so you can view the information contained in the error report before submitting it.

Behavior changes and deprecated features

Following is a list of changes to SQL Anywhere databases and database servers introduced in version 10.0.1, grouped by category.

Behavior changes

- **Changes to when intra-query parallelism is used** Intra-query parallelism is no longer used for connections with background_priority set to on. Additionally, intra-query parallelism is not used if the

number of server threads that are currently handling a request (ActiveReq server property) recently exceeded the number of CPU cores on the machine that the database server is licensed to use. See "Parallelism during query execution" [*SQL Anywhere Server - SQL Usage*].

A new server property, ExchangeTasksCompleted, returns the total number of internal tasks used for intra-query parallelism since the database server started. See "Database server properties" [*SQL Anywhere Server - Database Administration*].

- **Encryption results are no longer deterministic** Previously, encrypting values using the ENCRYPT function was deterministic. If you entered two identical input strings and two identical encryption keys, identical output data (ciphertext) was returned. Now, you can control whether encryption is deterministic using the new encrypt_aes_random_iv database option. The new default behavior is non-deterministic.

  Note
  Database servers that do not have this database option (version 10.0.0 and earlier) cannot decrypt data from databases that have this option set, even if it is set to Off.

- **ALTER DBSPACE RENAME attempts to open the dbspace if it was not open** Previously, if a database that uses dbspaces was started and one of its dbspaces could not be found, executing an ALTER DBSPACE ... RENAME statement for that dbspace updated the dbspace name in the catalog, but did not attempt to start the dbspace. Now, the database server attempts to open the dbspace after the catalog has been updated. See "ALTER DBSPACE statement" [*SQL Anywhere Server - SQL Reference*].

- **Changes to CREATE, ALTER, and DROP DBSPACE statements** The CREATE DBSPACE and DROP DBSPACE statements no longer accept the names of pre-defined dbspaces (SYSTEM, TEMPORARY, TEMP, TRANSLOG, and TRANSLOGMIRROR). If a user dbspace in a database created by an older version of the SQL Anywhere database server has the same name as one of the pre-defined dbspace names, the database server always refers to the user dbspace. See "Predefined dbspaces" [*SQL Anywhere Server - Database Administration*].

- **Changes to the output of the sa_conn_info system procedure** The output of the sa_conn_info system procedure has been changed to give more information about locks that connections are waiting on. The LockName field has been removed. Instead, two new fields, LockRowID and LockIndexID, have been added. If the connection is waiting on a lock that is associated with a particular row identifier, LockRowID contains that row identifier. If the connection is waiting on a lock that is associated with a particular index, LockIndexID contains the identifier of that index. See "sa_conn_info system procedure" [*SQL Anywhere Server - SQL Reference*].

  You must upgrade your database to use this feature. See "Upgrading version 10 and later databases" on page 315.

- **DBA permission no longer required for some system procedures** The sa_dependent_views, sa_get_dtt, sa_check_commit, and sa_materialized_view_info system procedures no longer require DBA permission to run.

- **Default unit of measure for CREATE DATABASE statement removed** When creating a database using the CREATE DATABASE statement, specifying the unit of measurement is no longer

optional if you specify a value for DATABASE SIZE. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **New maximum value for the default_timestamp_increment option** The maximum value for the default_timestamp_increment option is now 1000000 (1 second). See "default_timestamp_increment option" [*SQL Anywhere Server - Database Administration*].

- **dbdata10.dll removed** The functionality provided by the dbdata10 Dynamic Link Library has been incorporated in the SQL Anywhere .NET provider DLL. As a result, for Windows CE, there are now platform-specific versions of the SQL Anywhere .NET provider DLL.

- **Default cache size increased on NetWare** The default size of the database server cache on NetWare has been increased from 2 MB to 8 MB. See "-c dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Behavior changes resulting from client statement caching** The following behavior changes have been introduced as a result of the support for client statement caching:

  - An incorrect describe can occur for the same SQL statement that was described as having no result set, and then later does have a result set. For example:

    ```
    CREATE PROCEDURE p() NO RESULT SET BEGIN ... END
    Prepare, Describe, Drop "call p"
    ALTER PROCEDURE p() RESULT( ... ) BEGIN ... END
    Prepare, Describe, Drop "call p"  // describe returns no result set
    ```

  - When client statement caching is enabled and RememberLastStatement is enabled (-zl server option), the LastStatement property is the empty string when reusing a cached statement.

  - When client statement caching is enabled, if sa_get_request_times or sa_get_request_profile is used to process the request level log, the number of times a statement is executed may be incorrect. See "max_client_statements_cached option" [*SQL Anywhere Server - Database Administration*].

- **The Language utility (dblang) no longer requires administrator privileges** In previous versions of SQL Anywhere, users had to log in as an administrator to change language settings for localized versions of SQL Anywhere by using the dblang utility. This requirement has been removed.

- **Forward slashes in DISH service names no longer allowed** To prevent misinterpretation of DISH service names, forward slashes (/) are no longer permitted as part of the name for the service. See "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*].

- **Change to default value of SACommand.UpdateRowSource** Previously, the default value of SACommand.UpdatedRowSource was UpdatedRowSource.Both. It has been changed to UpdatedRowSource.OutputParameters. See "UpdatedRowSource property" [*SQL Anywhere Server - Programming*].

- **Change to the default value of the PrefetchRows connection parameter** When using the .NET Data Provider, the default value of the PrefetchRows connection parameter has changed from 10 to 200, to improve performance. The default value of SAConnectionStringBuilder.PrefetchRow has also been changed to 200. Prefetching is disabled if the

result set contains BLOB columns. See "PrefetchRows (PROWS) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Authenticated applications now use authenticate.sql instead of saopts.sql** In previous releases of the OEM Edition of SQL Anywhere, it was recommended that you store the authentication statement in the file *install-dir\scripts\saopts.sql*, so that it was applied whenever you created, rebuilt, or upgraded a database.

  It is now recommended that you store the authentication string in the file *install-dir\scripts\authenticate.sql*. See "Upgrading authenticated databases" [*SQL Anywhere Server - Database Administration*].

- **Long hostnames on HP-UX now accepted** Starting with the HP-UX 11i v2 September 2004 Update, system administrators could enable support for 255 byte hostnames by setting a kernel parameter. However, on SQL Anywhere servers on HP-UX computers with long hostname support enabled, the MachineName property and AppInfo HOST key returned at most 64 bytes of the hostname. Both MachineName and AppInfo can now return 255 byte hostnames.

- **iAnywhere JDBC driver URL header** In previous releases, when applications connected to SQL Anywhere using the iAnywhere JDBC driver, the URL passed to the JDBC driver began with the header **jdbc:odbc:**. Now, you can also begin the URL header with **jdbc:ianywhere:**. It is recommended that you use **jdbc:ianywhere:** to avoid conflicts with the Sun JDBC-ODBC bridge. See "Supplying a URL to the driver" [*SQL Anywhere Server - Programming*].

- **Retrieving a list of tables using jConnect when the Remarks value is longer than 128 CHARacters in length** Previously, if a JDBC application connected using jConnect and requested a list of tables, the results could be empty even though tables exist. This occurred when the string_rtruncation option was set to On, the application used the DatabaseMetaData.getTables method, and the Remarks value for any table was longer than 128 CHARacters. Now, Remarks values that are too long are truncated to 128 CHARacters, and the list of tables is returned. You must either run *jcatalog.sql*, or upgrade your database, to use this change. See "Installing jConnect system objects into a database" [*SQL Anywhere Server - Programming*] or "Upgrade utility (dbupgrad)" [*SQL Anywhere Server - Database Administration*].

- **Comparing CHAR and NCHAR values** For SQL Anywhere 10.0.0, combining CHAR and NCHAR domains resulted in an NCHAR comparison. However, this meant that applications upgraded to 10.0.0 could get different results, or experience performance degradation, when using host variables bound as SQL_C_WCHAR. In SQL Anywhere 10.0.0 variables bound as SQL_C_WCHAR are represented as NCHAR. In SQL Anywhere 10.0.1, new inference rules have been introduced to improve compatibility with existing applications, and to provide consistent, predictable results when combining CHAR and NCHAR domains. See "Comparisons between CHAR and NCHAR" [*SQL Anywhere Server - SQL Reference*].

- **Asynchronous I/O disabled on Linux kernels earlier than 2.6.12** Because of a bug in Linux kernels earlier than 2.6.12, when you run a SQL Anywhere database server on one of the affected kernels, asynchronous I/O is disabled by default. If you want to use asynchronous I/O, then you must upgrade your kernel to 2.6.12 or later.

- **OEM Edition documentation moved** In previous releases of the SQL Anywhere OEM Edition, instructions for setting up authenticated applications were included in a separate *.pdf* or *.html* file. Now, this information is available in the following locations:

  - "Running authenticated SQL Anywhere applications" [*SQL Anywhere Server - Database Administration*]
  - "connection_authentication option" [*SQL Anywhere Server - Database Administration*]
  - "database_authentication" [*SQL Anywhere Server - Database Administration*]

- **Unload utility -e and -t options no longer require case sensitive table names for case sensitive databases** In previous releases, when you unloaded a case sensitive database using the dbunload utility with the -e or -t options, these options required case sensitive table names. Now, the table names are now case insensitive.

- **Loading data into temporary tables** The behavior when loading data into temporary tables has changed. With the exception of a LOCAL TEMPORARY TABLE that is defined with ON COMMIT DELETE ROWS, a commit is now automatically performed before and after performing a LOAD TABLE on a temporary table. If the load fails, all rows in the temporary table are now removed, including rows which were present before the load.

  For of a LOCAL TEMPORARY TABLE with ON COMMIT DELETE ROWS, there is no change in behavior; no commits are performed. This means that in the event of a partial load due to a failure during the load, this type of temporary table would contain only some of the loaded rows, and may also be missing other rows that were present before the load.

  Also, loading into a temporary table now fails if the table contains rows referenced by the foreign key of another table.

  You cannot load into a GLOBAL TEMPORARY TABLE that is defined with ON COMMIT DELETE ROWS.

- **Default case sensitivity for Japanese databases using the UCA collation** The default case and accent sensitivity of UCA collations when creating a Japanese database is now *sensitive*. A Japanese database is defined as any database created on a Japanese computer where the OS language or CHARacter set is Japanese, or any database created with a Japanese CHAR collation, such as 932JPN, or EUC_JAPAN.

  The default case sensitivity of UCA collations when creating a non-Japanese database is still *insensitive*.

  The defaults for case and accent sensitivity can still be overridden using the dbinit -c and -a (or -c- and -a-) options, respectively, by using collation tailoring syntax, or by using the CASE and ACCENT clauses of the CREATE DATABASE STATEMENT. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*] and "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

Deprecated and discontinued features

- **SQL Flagger support for the SQL/1992 standard** SQL Flagger support for SQL:1992 (all levels) is deprecated.

- **dbinit -e option is deprecated** The dbinit -e option, used for specifying simple encryption when creating a database, has been deprecated. Use the -ea option (specifically, -ea simple) to specify simple encryption. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*].

- **SADbType.oldbit data type removed** The SADbType.oldbit enumeration constant has been removed from the SQL Anywhere .NET provider.

- **-gx server option deprecated** On Windows desktop platforms, the database server scheduler now attempts to maintain the affinity of requests so it can use CPU cache. As a result, requests run on one CPU as much as possible. As well, the -gx server option, which was used for specifying the number of operating system threads for the database server to use has been deprecated. This option is now ignored by the database server.

- **CASE and ACCENT clauses of CREATE DATABASE statement deprecated** As a result of the addition of collation tailoring support using the COLLATION and NCHAR COLLATION of the CREATE DATABASE STATEMENT, the CASE and ACCENT clauses of this statement are deprecated. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

MobiLink

The following sections describe the new features, behavior changes, and deprecated features in MobiLink for version 10.0.1.

New features

MobiLink server

- **New ODBC driver for Oracle** There is now an ODBC driver for Oracle called iAnywhere Solutions 10 - Oracle that is custom-tailored for MobiLink applications.

  See "New ODBC driver for Oracle" on page 180.

- **Restructured encryption layer** The MobiLink encryption layer has been restructured and improved. This change is transparent and does not require any changes to applications.

- **MobiLink Server Log File Viewer** A new window has been added that allows you to view MobiLink log files. The Log File Viewer provides enhanced functionality such as filtering logged information and viewing summaries and statistics.

  See "MobiLink server Log File Viewer" [*MobiLink - Server Administration*].

- **Improved memory use** On Windows, the MobiLink server (a 32-bit process) will now use more memory if required. Previously, it was limited to less than 2 GB, but it can now use significantly more

memory if more is required and available. Use the mlsrv10 -cm option to set the maximum size for the server memory cache. More memory can lead to less paging to disk, which can improve performance.

MobiLink plug-in to Sybase Central

- **New functionality in Model mode**
  - **Table Mapping** It is easier to create and change table and column mappings. You can enable a table mapping by selecting from a dropdown list in the Mapping Direction column for a table. Similarly, you now specify whether a column is mapped in the Column Mappings tab. The New Table Mappings wizard has been removed.

    See "Modifying table and column mappings" [*MobiLink - Getting Started*].
  - **Create new remote tables** It is easier to create new remote tables based on the consolidated database schema. You can create a new remote table with the same name and columns as a table in the consolidated database using the Create New Remote Tables window. This window also maps the remote database tables to the consolidated tables in the model.
  - **Delete remote database tables and columns** You can now delete remote database tables and columns in the Mappings tab. You can remove a remote table or column from the remote database schema in the model by selecting the table or column and choosing Edit » Delete.

MobiLink clients

- **More convenient way to specify network name on Windows CE** You can now specify the keywords default_internet or default_work as the name in the network_name protocol option so that your default setting is the name that is used.

  See "network_name" [*MobiLink - Client Administration*].
- **Enhanced functionality for delete_old_logs** The database option delete_old_logs now allows you to specify a number of days. Logs are then deleted that were created after that time period.

  See "delete_old_logs option [SQL Remote]" [*SQL Anywhere Server - Database Administration*].
- **New hook** The new hook sp_hook_dbmlsync_set_ml_connect_info allows you to set the network protocol and network protocol options immediately before dbmlsync attempts to connect to the MobiLink server.

  See "sp_hook_dbmlsync_set_ml_connect_info" [*MobiLink - Client Administration*].

Security

There are two new utilities called createcert and viewcert for managing transport-layer security. See:

- "Transport-layer security" on page 180
- "Certificate utilities" [*SQL Anywhere Server - Database Administration*]

Behavior changes and deprecated features

Following is a list of changes to MobiLink introduced in version 10.0.1.

MobiLink plug-in to Sybase Central

- **Add Table Mappings wizard is removed** New functionality has been added to Model mode to add new tables to the remote database schema.

  See "Modifying table and column mappings" [*MobiLink - Getting Started*].

- **Default dbmlsync batch file improved** When you deploy a MobiLink model, you create a file called *model-name_dbmlsync.bat*. Previously, the default dbmlsync command in this file included the -qc option, which closed the dbmlsync window after synchronization. This meant that it was difficult to determine whether the synchronization was successful. The -qc option is now removed from the default dbmlsync command line.

Security

The TLS utilities readcert, gencert, and reqtool have been deprecated. They are replaced with utilities called createcert and viewcert. See:

- "Certificate utilities" [*SQL Anywhere Server - Database Administration*]

QAnywhere

The following sections describe the new features, behavior changes, and deprecated features in QAnywhere for version 10.0.1.

New features

Following is a list of additions to QAnywhere introduced in version 10.0.1.

- **Dynamic addressing** The QAnywhere Agent can now detect active networks and automatically adjust the communication protocol and address of the MobiLink server without restarting.

  See "-xd qaagent option" [*QAnywhere*].

- **Maximum download size** You can now set a maximum size for the download of messages.

  See "-idl qaagent option" [*QAnywhere*] and ias_MaxDownloadSize in "Predefined client message store properties" [*QAnywhere*].

- **QAnywhere Server Log File Viewer** A new viewer has been added that allows you to view QAnywhere server log files. The Log File Viewer provides enhanced functionality such as filtering logged information and viewing summaries and statistics.

See "Logging the QAnywhere server" [*QAnywhere*].

Client API enhancements

- **Ability to handle exceptions during processing for a message listener in the .NET API** ExceptionListener delegates have been added to the .NET API. This functionality already exists in the Java API.

  See:

  - "ExceptionListener delegate" [*QAnywhere*]
  - "ExceptionListener2 delegate" [*QAnywhere*]

- **Ability to pass the owning QAManager of a message to the Listener** New interfaces have been added to the .NET and Java APIs that are convenient for calling QAManagerBase API calls inside the Listener. This is useful, for example, when acknowledging a message. The new interfaces do not require you to reference a global instance of the QAManagerBase or use other coding techniques to pass the QAManagerBase into the Listener.

  See:

  - .NET API: "MessageListener2 delegate" [*QAnywhere*]
  - Java API: "QAMessageListener2 interface" [*QAnywhere*]

Behavior changes and deprecated features

Following is a list of changes to QAnywhere introduced in version 10.0.1.

- **Client message store transaction log** By default, the contents of the client message store transaction log are now deleted at checkpoints.

  See "-m dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*].

  You can change this behavior by specifying the StartLine parameter in the qaagent -c option.

  See "-c qaagent option" [*QAnywhere*].

SQL Remote

The following sections describe behavior changes and deprecated features in SQL Remote for version 10.0.1.

Behavior changes and deprecated features

Following is a list of changes to SQL Remote introduced in version 10.0.1.

- **VIM and MAPI deprecated** Support for the VIM and MAPI message systems is deprecated in this release. The file, ftp, and SMTP message types are still supported. See "SQL Remote message systems" [*SQL Remote*].

UltraLite

The following sections describe the new features, behavior changes, and deprecated features in UltraLite for version 10.0.1.

New features

Following is a list of additions to UltraLite introduced in version 10.0.1.

- **New platforms and devices** Windows Vista is now supported in this release.
- **Improved SQL performance** Historically, UltraLite would default to using the primary key index if, the UltraLite query optimizer determined that there was not any benefit to using an existing index for the query.

  With this version, instead of using the primary key, UltraLite will now access rows directly from the database pages. When this happens, you can expect to see query results returned in a different order from previous releases, because rows are no longer ordered by the primary key index. If the ordering of data is a concern, use an ORDER BY clause in your query. See "Using direct page scans" [*UltraLite - Database Management and Reference*] and "Using index scans" [*UltraLite - Database Management and Reference*].
- **Database property and option accessibility via SQL** Previous versions only allowed you to access database properties and options with methods from each UltraLite API. Now, UltraLite SQL introduces the following statement and function, so that you can set and retrieve properties and options via SQL (including Interactive SQL):
  - the SET OPTION statement. See "SET OPTION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].
  - the DB_PROPERTY function. See "DB_PROPERTY function [System]" [*UltraLite - Database Management and Reference*].
- **Increased number of concurrent connections** UltraLite now supports more concurrent connections. A single database still supports 14 connections. However, the number of overall concurrent database connections have increased:
  - 8 databases and 16 concurrent connections for Palm OS and Symbian OS.
  - 32 databases and 64 concurrent connections for all remaining platforms.

SQLCA restrictions can further limit connections
The total number of SQLCAs you can use to connect to the engine is 31. Keep this number in mind knowing that some components use one SQLCA per database manager and one SQLCA per connection. That means, for example, if your application is using the UltraLite.NET API, your real connection limit is reduced to a total of 30 connections only.

- **Commit flush operations** Historically, any commit operation performed with either the COMMIT statement or API call would only complete after UltraLite flushed a transaction safely to storage. With this release, you can now configure these behaviors and logically separate them as distinct operations:

  - A logical commit now gives you the ability to roll transactions back in an application.
  - A checkpoint now gives you a recovery point after a failure. This allows you to recover to the last committed transaction that has been flushed to memory.

  However, you can also enhance the performance of UltraLite applications that use the autocommit feature —particularly if you use group commit flushes. See "Backing up and recovering data in UltraLite" [*UltraLite - Database Management and Reference*] and "Flushing single or grouped transactions" [*UltraLite - Database Management and Reference*].

  This new behavior is supported with the following features:

  - The CHECKPOINT statement. See "CHECKPOINT statement [UltraLite]" [*UltraLite - Database Management and Reference*].

    The Checkpoint methods for the UltraLite embedded SQL API and C++ API components. Other languages must use the CHECKPOINT statement instead. See:

    - UltraLite Embedded SQL: "ULCheckpoint method" [*UltraLite - C and C++ Programming*]
    - UltraLite C++: "Checkpoint method" [*UltraLite - C and C++ Programming*]

  - The COMMIT_FLUSH connection parameter. See "UltraLite COMMIT_FLUSH connection parameter" [*UltraLite - Database Management and Reference*].
  - The commit_flush_timeout and commit_flush_count database options. See "UltraLite commit_flush_timeout option [temporary]" [*UltraLite - Database Management and Reference*] and "UltraLite commit_flush_count option [temporary]" [*UltraLite - Database Management and Reference*].

Behavior changes and deprecated features

Following is a list of changes to UltraLite introduced in version 10.0.1.

- **ulafreg** ulafreg has been changed so standard output is no longer available. You still run this utility from the command line with the options you require. However, you must now click **Edit** » **Copy** to copy the output to the clipboard. You can then save it to file using a text editor of your choosing.

- **FIPS support on Windows CE** You no longer need to run the following executable to support FIPS on Windows CE devices: *install-dir\ultralite\ce\arm\fips\setup.exe*. Now, you simply deploy the following file: *install-dir\ultralite\ce\arm\sbgse2.dll*.

Product-wide features

The following sections describe the new features, behavior changes, and deprecated features that affect all components of SQL Anywhere version 10.0.1.

New features

Following is a list of product-wide additions introduced in version 10.0.1.

New ODBC driver for Oracle

There is now a native ODBC driver for Oracle called iAnywhere Solutions 10 - Oracle.

Previously, iAnywhere rebranded an Oracle driver that was created by a third party. Now, iAnywhere has its own Oracle ODBC driver for use by SQL Anywhere applications. Using this driver, you can expect faster bug fixes and improved functionality when dealing with international CHARacter data. If you have MobiLink deployments using an Oracle consolidated database or you use remote data access to connect to Oracle, you are strongly urged to switch to this new driver.

See "iAnywhere Solutions 12 - Oracle ODBC driver" [*MobiLink - Server Administration*].

Transport-layer security

- **New certificate utilities** Two new utilities, createcert and viewcert, allow you to make, modify, and view security certificates. Previously, the utilities gencert, reqtool, and readcert were used for this purpose (those utilities have been deprecated).

  Viewcert allows you to view several types of PKI object. Previously, you could only view certificates. Viewcert also allows you to view both PEM and DER objects; previously, you could only view PEM objects. With viewcert you can also convert between PEM and DER, as well as encrypt or decrypt passwords.

  Createcert combines the functionality of the old gencert and reqtool utilities, and provides new functionality. When creating an ECC curve, you can now choose your curve; previously you had to use sect163k1. You can use key sizes from 512 to 16384 bits; previously, the size limit was 512 to 2048 bits. There is now a default GUID serial number; previously there was no default. You can now optionally create certificates that can sign other certificates. You can also specify advanced options that determine how a certificate's private key can be used. Finally, you can now use unencrypted private keys; previously, all private keys had to have a password.

  See "Certificate Creation utility (createcert)" [*SQL Anywhere Server - Database Administration*] and "Certificate Viewer utility (viewcert)" [*SQL Anywhere Server - Database Administration*].

- **Upgraded Certicom Security Builder for FIPS support on Windows CE** SQL Anywhere products can use one of two FIPS-certified modules for encryption, both from Certicom:

- On Palm OS, you must still use Security Builder Government Services Edition v1.0.1.
- On Windows CE, you must now use Security Builder Government Services Edition v2.0.0 because these libraries can be signed for use on Windows Mobile.

See "FIPS-approved encryption technology" [*SQL Anywhere Server - Database Administration*].

Behavior changes

Following is a list of product-wide changes in version 10.0.1.

Transport-layer security

- **gencert, readcert, and reqtool are deprecated** The security utilities gencert, reqtool, and readcert are deprecated. Gencert and reqtool are replaced by createcert. Readcert is replaced by viewcert.

  See "Certificate utilities" [*SQL Anywhere Server - Database Administration*].

Licensing

- **Server licensing information is now stored in .lic files** In previous releases, licensing information for the SQL Anywhere personal database server, the SQL Anywhere network database server, and the MobiLink server was stored in the server executable. This information is now stored in a *.lic* file that is located in the same directory as the server executable. If the *.lic* cannot be found for an executable, then the executable does not start.

  As a result of this change, the exename member of the a_dblic_info structure now specifies the executable or license file name.

  See:

  - "Server Licensing utility (dblic)" [*SQL Anywhere Server - Database Administration*]
  - "Deploying database servers" [*SQL Anywhere Server - Programming*]
  - "Deploying the MobiLink server" [*MobiLink - Server Administration*]
  - "a_dblic_info structure" [*SQL Anywhere Server - Programming*]

Windows Vista support issues

SQL Anywhere version 10.0.1 supports the Windows Vista operating system. Following are some issues relating to running SQL Anywhere software on Vista:

- **Windows Vista security** Windows Vista incorporates a new security model. User Account Control (UAC) is enabled by default and may affect the behavior of programs that expect to be able to write files, especially when the computer supports more than one user. Depending on where and how files and directories are created, a file created by one user may have permissions that do not allow another user to read or write to that file. If you install SQL Anywhere into the default directories, files and directories that require read/write access for multiple users are set up appropriately.

- **SQL Anywhere elevated operations agent** In Vista, certain actions require privilege elevation to execute when run under User Account Control. The following programs may require elevation in SQL Anywhere: *dbdsn.exe*, *dbelevate10.exe*, *dblic.exe*, *dbsvc.exe*, *installULNet.exe*, *mlasinst.exe*, *SetupVSPackage.exe*, *ulcond10.exe*, and *ulafreg.exe*.

  The following DLLs require elevation when they are registered or unregistered: *dbctrs10.dll*, *dbodbc10.dll*, *dboledb10.dll*, and *dboledba10.dll*.

  On a Vista system with User Account Control activated, you may receive an elevation prompt for the SQL Anywhere elevated operations agent. The prompt is issued by the Vista User Account Control system to confirm that you want to continue running the identified program (if logged on as an administrator) or to provide administrator credentials (if logged on as a non-administrator).

- **Deployment changes** The program *dbelevate10.exe* is used internally by SQL Anywhere components to perform operations that require elevated privileges. This executable must be included in deployments of SQL Anywhere.

- **ActiveSync support** The Microsoft ActiveSync utility is not supported in Vista. It is replaced by the Windows Mobile Device Center. You can use the SQL Anywhere ActiveSync Provider Installation utility with Windows Mobile Device Center.

- **SQL Anywhere executables signed** SQL Anywhere executables on Vista are signed by iAnywhere Solutions, Inc.

- **New license files** The installation of 10.0.1 includes procedures that create new license files for SQL Anywhere. License information from an existing installation is extracted from the old location within the executable files and moved to the new location (*dbsrv10.lic*, *dbeng10.lic*, and *mlsrv10.lic* files in the same directory as the executables).

  See "Server Licensing utility (dblic)" [*SQL Anywhere Server - Database Administration*].

- **Samples** Samples now correctly handle SQL Anywhere installation path names that contain one or more spaces.

- **Windows services** Vista-compliant services are not allowed to interact with the desktop. On Windows Vista, no SQL Anywhere services interact with the desktop (even if **Allow Service To Interact With Desktop** is enabled in the service definition). SQL Anywhere database servers can be monitored using the dbconsole utility or from Sybase Central.

  Sybase Central disables the option to allow service to interact with desktop when running on Windows Vista.

- **PowerDesigner, InfoMaker, and DataWindow .NET** The PowerDesigner, InfoMaker, and DataWindow .NET components included with SQL Anywhere are not officially supported on Windows Vista. As a result, you may experience issues running these components in a Vista environment. Refer to the respective product documentation for instructions on how to run in a Vista environment, as well as how to obtain a Vista-supported version of these products.

What's new in version 10.0.0

Deprecated feature lists subject to change
As with all forward-looking statements, the lists of deprecated features are not guaranteed to be complete and are subject to change.

SQL Anywhere

MobiLink

QAnywhere

SQL Remote

UltraLite

Sybase Central and Interactive SQL

Documentation enhancements

SQL Anywhere

The following sections describe the new features, behavior changes, and deprecated features in SQL Anywhere for version 10.0.0.

New features

Following is a list of additions to SQL Anywhere databases and database servers introduced in version 10.0.0.

Main features

- **Support for parallelism to improve performance** The database server now supports the use of multiple processors for processing a single query. Intra-query parallelism is beneficial when the number of simultaneously executing queries is less than the number of available processors. See "Parallelism during query execution" [*SQL Anywhere Server - SQL Usage*].

- **Support for database mirroring** SQL Anywhere now supports database mirroring, which is a mechanism to increase the availability of a database. It involves using either two or three database servers running on separate computers and communicating with each other in either synchronous or asynchronous mode. See "Introduction to database mirroring" [*SQL Anywhere Server - Database Administration*].

  The following features have been added to support database mirroring:

  - "synchronize_mirror_on_commit option" [*SQL Anywhere Server - Database Administration*]
  - Alternate server names for database servers. See "-sn dbsrv12 database option" [*SQL Anywhere Server - Database Administration*] and "START DATABASE statement" [*SQL Anywhere Server - SQL Reference*].
  - ServerName property
  - AlternateServerName property
  - RetryConnectionTimeout property
  - ALTER DATABASE *dbname* FORCE START. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*].
  - MirrorServerDisconnect and MirrorFailover system events. See "Understanding system events" [*SQL Anywhere Server - Database Administration*].
  - "-xf dbsrv12 server option" [*SQL Anywhere Server - Database Administration*]
  - "-xp dbsrv12 database option" [*SQL Anywhere Server - Database Administration*]
  - A new trap for the SQL Anywhere SNMP Extension Agent. See "Using traps" [*SQL Anywhere Server - Database Administration*].
  - MirrorServerName parameter added to the EVENT_PARAMETER function. See "EVENT_PARAMETER function [System]" [*SQL Anywhere Server - SQL Reference*].

  In addition to database mirroring, SQL Anywhere now provides Veritas Cluster Server agents for both databases (SADatabase agent) and database servers (SAServer agent). See "Using the SQL Anywhere Veritas Cluster Server agents" [*SQL Anywhere Server - Database Administration*].

- **Support for snapshot isolation** When you use snapshot isolation, the database keeps a copy of the original data while a user is changing it, and makes the original data available to other users who want to read it. Snapshot isolation is completely transparent to users, and can help reduce deadlocks and lock contentions. See "Snapshot isolation" [*SQL Anywhere Server - SQL Usage*].

  The following features have been added or enhanced to support snapshot isolation:

  - "allow_snapshot_isolation option" [*SQL Anywhere Server - Database Administration*]
  - "isolation_level option" [*SQL Anywhere Server - Database Administration*]
  - "sa_snapshots system procedure" [*SQL Anywhere Server - SQL Reference*]
  - "sa_transactions system procedure" [*SQL Anywhere Server - SQL Reference*]
  - LockCount, SnapshotCount, SnapshotIsolationState, and VersionStorePages database properties
  - allow_snapshot_isolation, LockCount, and SnapshotCount connection properties
  - Version Store Pages Performance Monitor statistic
  - "SET statement [T-SQL]" [*SQL Anywhere Server - SQL Reference*]
  - "OPEN statement [ESQL] [SP]" [*SQL Anywhere Server - SQL Reference*]
  - "The ValuePtr parameter" [*SQL Anywhere Server - SQL Usage*]

- **Support for application profiling and diagnostic tracing** Existing application profiling capabilities, such as stored procedure profiling and request logging, have been integrated into a single, unified interactive interface the SQL Anywhere plug-in for Sybase Central. When you profile your application from Sybase Central, recommendations are provided to help you improve database performance.

  For more information about application profiling in Sybase Central, see "Application profiling" [*SQL Anywhere Server - SQL Usage*].

- **Support for materialized views** To improve performance in environments where the database is large and frequent queries result in repetitive aggregation and join operations on large amounts of data, SQL Anywhere now supports materialized views. See "Working with materialized views" [*SQL Anywhere Server - SQL Usage*].

  The database server has been enhanced to automatically decide, based on cost, which materialized views can be used to answer parts of a query instead of using base tables referenced directly by the query. See "Improving performance with materialized views" [*SQL Anywhere Server - SQL Usage*].

  Two new system tables, ISYSMVOPTION and ISYSMVOPTIONNAME, have been added to store information about materialized views. See "SYSMVOPTION system view" [*SQL Anywhere Server - SQL Reference*] and "SYSMVOPTIONNAME system view" [*SQL Anywhere Server - SQL Reference*].

- **Support for NCHAR data** SQL Anywhere now supports the NCHAR data type. NCHAR data types are used for storing Unicode character data. See "NCHAR data type" [*SQL Anywhere Server - SQL Reference*].

The following new functions have been added in support of NCHAR:

- "UNISTR function [String]" [*SQL Anywhere Server - SQL Reference*]
- "CONNECTION_EXTENDED_PROPERTY function [String]" [*SQL Anywhere Server - SQL Reference*]
- "UNICODE function [String]" [*SQL Anywhere Server - SQL Reference*]
- "NCHAR function [String]" [*SQL Anywhere Server - SQL Reference*]
- "TO_CHAR function [String]" [*SQL Anywhere Server - SQL Reference*]
- "TO_NCHAR function [String]" [*SQL Anywhere Server - SQL Reference*]

The following functions SORTKEY and COMPARE functions have new parameters to support the NCHAR data type:

- "SORTKEY function [String]" [*SQL Anywhere Server - SQL Reference*]
- "COMPARE function [String]" [*SQL Anywhere Server - SQL Reference*]

SQL Anywhere now properly sorts multibyte character sets when using the Unicode Collation Algorithm (UCA).

The Initialization utility (dbinit) and Unload (dbunload) utilities also have new options to support the NCHAR data type. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*] and "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere now uses International Components for Unicode (ICU) for Unicode support. See "International languages and character sets" [*SQL Anywhere Server - Database Administration*].

To support ICU and the handling of NCHAR data, the following property changes have been made:

- A new NcharCharSet database and connection extended property has been added. This property returns the NCHAR character set in use by the database or connection.
- A new AccentSensitive database property has been added. This property returns the status of the accent sensitivity feature.
- The CharSet database and connection properties are now extended properties.

See "Database properties" [*SQL Anywhere Server - Database Administration*] and "Connection properties" [*SQL Anywhere Server - Database Administration*].

- **Internal performance enhancements** To improve database server performance, virtual machine technology has been used to re-architect the representation and evaluation of SQL expressions, which significantly improves throughput.
- **Support for view dependencies** The catalog now stores information about the dependencies of views. Specifically, the catalog keeps track of the views, tables and columns upon which each view in the database depends. When you make an alteration to an object upon which a view depends, the database server automatically performs additional operations to ensure that the view definition is not left in a state where it could return incorrect results. See "View dependencies" [*SQL Anywhere Server - SQL Usage*].

Two new tables, ISYSDEPENDENCY and ISYSOBJECT, have been added to store information about system objects and their dependencies. See "SYSDEPENDENCY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSOBJECT system view" [*SQL Anywhere Server - SQL Reference*].

- **Improved checkpoint algorithm** The database server can now initiate a checkpoint and perform other operations while the checkpoint takes place. Previously, all activity would stop while the checkpoint took place. If a checkpoint is already in progress, then any operation like an ALTER TABLE or CREATE INDEX that initiates a new checkpoint must wait for the current checkpoint to finish. See "Understanding the checkpoint log" [*SQL Anywhere Server - Database Administration*].

- **Locking enhancements** The following enhancements have been made to locking:

  - **Classes of locks** SQL Anywhere now supports four distinct classes of locks: schema locks, table locks, row locks, and position locks. The sa_locks system procedure has been modified to more clearly document the types of locks that each transaction holds to permit more accurate analysis of locking issues. See "How locking works" [*SQL Anywhere Server - SQL Usage*] and "sa_locks system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **Support for intent locks** A new type of lock, called an intent lock, has been introduced for both table locks and row locks. Intent locks are used by an application to signal its intention to update a table or a set of rows within that table. Intent locks are now acquired when an application uses SELECT FOR UPDATE or FETCH FOR UPDATE statements (or their equivalent constructions in various programming interfaces). Intent locks block other intent locks and write locks, but do not block read locks. This support gives a higher degree of concurrency to those applications that use locking as an explicit concurrency control mechanism. See "Using cursors" [*SQL Anywhere Server - Programming*] and "Intent locks" [*SQL Anywhere Server - SQL Usage*].

  - **Key range locking eliminated in some situations** Changes to index maintenance algorithms now permit the database server to place write locks on individual index entries, rather than on a range of keys. This will improve concurrency and eliminate unnecessary blocking due to concurrent INSERT operations in a variety of circumstances. See "How locking works" [*SQL Anywhere Server - SQL Usage*].

- **Indexing enhancements** The following enhancements have been made to indexing:

  - **New index implementation** Previous releases of SQL Anywhere contained two different indexing implementations that were chosen automatically based on the declared size of the indexed columns. In SQL Anywhere 10, a new implementation of compressed B-tree indexes is used throughout, and older B-tree indexing technology has been eliminated. The new indexes store a compressed form of the index key value in the index entry, separate and distinct from the value in the row. This is required to support snapshot isolation.

  - **Support for snapshot isolation** In previous SQL Anywhere releases, index entries were deleted on UPDATE or DELETE statements immediately. To support snapshot isolation, there is potential for several index entries to point to the same logical row with different index key values. These multiple index entries are managed by the database server so that any one connection can see only one of the entries for any given row. Periodically, a daemon within the server will physically delete these extra index entries when they are no longer needed (as transactions

COMMIT or ROLLBACK). Retaining index entries for uncommitted DELETEs also improves semantic consistency of the concurrency control mechanisms in SQL Anywhere. See "Snapshot isolation" [*SQL Anywhere Server - SQL Usage*].

- **Improved BLOB storage control and performance** You can now control the amount of a BLOB value that is stored in a table row (inline). You can also control whether to index BLOB values. These enhancements improve searching through, and accessing, BLOBs, and are made available through three new clauses in the CREATE TABLE and ALTER TABLE statements: INLINE, PREFIX, and [NO] INDEX. BLOB values can now be shared within, or among rows of the same table, reducing storage requirements by eliminating the need to store duplicate BLOB values. See "Storing BLOBs" [*SQL Anywhere Server - Database Administration*], "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*], and "ALTER TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **Support for column compression** You can now compress individual columns in a table. Compression is achieved using the deflate compression algorithm. This is the same compression used by the COMPRESS function, and is also the algorithm used in Windows *.zip* files. See "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*] and "ALTER TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **Support for table encryption** Instead of encrypting an entire database to secure data, you can now encrypt individual tables in the database. Table encryption must be enabled in the database when it is initialized. See "Table encryption" [*SQL Anywhere Server - Database Administration*].

Database connections

- **Support for single or double quotes** Values in connection strings can now be enclosed in single or double quotes. This allows characters such as spaces and semicolons to be used in connection string values. See "Connection parameters passed as connection strings" [*SQL Anywhere Server - Database Administration*].

- **Connection strings now allow T, Y, F, and N as boolean values** You can now specify T or Y to indicate true, and F or N to indicate false, when specifying connection parameters and protocol options in connection strings. See "Connection parameters" [*SQL Anywhere Server - Database Administration*].

- **Some connection strings and protocol options now accept values with k, m, and g suffixes** The following connection parameters and protocol options now accept k, m, and g as suffixes indicating kilobytes, megabytes, and gigabytes, respectively:

  - "CommBufferSize (CBSIZE) connection parameter" [*SQL Anywhere Server - Database Administration*]
  - "CompressionThreshold (COMPTH) connection parameter" [*SQL Anywhere Server - Database Administration*]
  - "PrefetchBuffer (PBUF) connection parameter" [*SQL Anywhere Server - Database Administration*]
  - "LogMaxSize (LSIZE) protocol option" [*SQL Anywhere Server - Database Administration*]
  - "MaxRequestSize (MAXSIZE) protocol option" [*SQL Anywhere Server - Database Administration*]
  - "ReceiveBufferSize (RCVBUFSZ) protocol option" [*SQL Anywhere Server - Database Administration*]
  - "SendBufferSize (SNDBUFSZ) protocol option" [*SQL Anywhere Server - Database Administration*]

- **AppInfo returns IP address for Windows clients** In previous releases, the AppInfo connection parameter only returned the IP address of the client computer on Unix and NetWare clients. The IP address is now returned for Windows clients as well. See "AppInfo (APP) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Auditing individual connections** The conn_auditing temporary database option allows you to enable or disable auditing for a specific connection when the option is set in a login procedure. The auditing database property has been added to help you obtain information about the auditing status of a database. See "conn_auditing option" [*SQL Anywhere Server - Database Administration*].

- **RetryConnectionTimeout connection parameter** The RetryConnectionTimeout (RetryConnTO) connection parameter tells the client library to retry the connection attempt, as long as the server is not found, for the specified period of time. See "RetryConnectionTimeout (RetryConnTO) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Support for IPv6** IPv6 is supported now on Windows, Linux, Mac OS X, Solaris, AIX, and HP-UX. Servers running on these operating systems now listen on all available IPv4 and IPv6 addresses, and anywhere you can specify an IP address on the client or server (such as the HOST=, MYIP=, and BROADCAST= TCP protocol options), you can now specify an IPv6 address. See "IPv6 support in SQL Anywhere" [*SQL Anywhere Server - Database Administration*].

- **New parameters for LDAP registration** The read_authdn and read_password parameters can be used to register the database server with LDAP if the database server is an Active Directory server. See "Connecting using an LDAP server" [*SQL Anywhere Server - Database Administration*].

Backup and recovery

- **Checksums calculated automatically for critical database pages** The database server records checksums for critical database pages, regardless of whether checksums are enabled for the database. As a result, you may see warnings about checksum violations when you validate your database, even if the database does not have checksums enabled. Additionally, the database server

shuts down with a fatal error when it tries to access a corrupt critical page. See "Validating checksums" [*SQL Anywhere Server - Database Administration*].

- **Applying multiple transaction logs at startup for recovery** By default, you must apply transaction logs individually, in the correct order when recovering a database. When the new -ad, -ar, and -as recovery options specified when starting the database server, you do not need to manually specify the order in which transaction logs are applied to the database. Because the database server and the database are running while the transaction logs are applied, the server's cache remains in a warm state, reducing total recovery time. See "-ad dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*], "-ar dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*], and "-as dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*].

- **Support for parallel database backups** The SQL Anywhere database server now supports parallel backups for server-side image backups. Parallel database backups take advantage of physical I/O to perform read and write information in parallel, instead of sequentially, which improves performance. You can perform parallel backups in any of the following ways:

  - "Backup utility (dbbackup)" [*SQL Anywhere Server - Database Administration*]
  - "BACKUP statement" [*SQL Anywhere Server - SQL Reference*]
  - "db_backup function" [*SQL Anywhere Server - Programming*]

- **Tracking information on the last backup** A new column, LAST_BACKUP, has been added to the ISYSHISTORY system table to store information about the last backup. See "SYSHISTORY system view" [*SQL Anywhere Server - SQL Reference*].

Security

This section explains the enhancements made to SQL Anywhere to improve security.

- **RSA now included with SQL Anywhere** You no longer have to purchase a separate license to use RSA encryption. See "Separately licensed components" [*SQL Anywhere 12 - Introduction*].

- **Enhancements to FIPS support** The following FIPS-related changes have been made to the database server:

  - The FIPS DLL has been renamed to *dbfips10.dll*. In version 9.0, it was called *dbrsa9f.dll*.

  - The HASH function now accepts two new algorithms: SHA1_FIPS and SHA256_FIPS. These are the same as the SHA1 and SHA256 algorithms, but are the FIPS-validated Certicom versions.

  - If the -fips server option is specified and a non-FIPS algorithm is given to the HASH function, the database server uses SHA1_FIPS instead of SHA1, SHA256_FIPS instead of SHA256, and returns an error if MD5 is used (MD5 is not a FIPS algorithm).

  - If the -fips option is specified, the database server uses SHA256_FIPS for password hashing.

Also, the -fips option and FIPS functionality are now available on more platforms. To see the list of platforms on which the -fips option is supported, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

- **Kerberos authentication** SQL Anywhere now supports Kerberos authentication. Kerberos authentication lets you use your Kerberos credentials to connect to the database without specifying a user ID or password. See "Kerberos authentication" [*SQL Anywhere Server - Database Administration*].

- **New authorities added** The following authorities have been added:

  - **BACKUP authority** You can assign BACKUP authority to a user so that they can perform backups, instead of granting the user DBA authority. See "BACKUP authority" [*SQL Anywhere Server - Database Administration*].

  - **VALIDATE authority** A new authority for validation operations, VALIDATE, has been added. VALIDATE authority is required to perform the operations executed by the different VALIDATE statements, such as database, table, index, and checksum validation. See "VALIDATE authority" [*SQL Anywhere Server - Database Administration*].

- **Securing features for a database server** The -sf database server option lets you specify features, or groups of features, that are secured (disabled) for databases running on the database server. See "-sf dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

  The -sk server option lets you specify a key that can be used to enable disabled features when used with the secure_feature_key database option. You can also change the set of disabled features using the sa_server_option system procedure SecureFeatures property. See "-sk dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Database utilities

- **@filename can be reused for several utilities** Command parameter files can now be selectively parsed for the utility using the parameter file. The parsing is based on simple conditional directives placed in the parameter file. See "Using conditional parsing in configuration files" [*SQL Anywhere Server - Database Administration*].

- **Data Source utility (dbdsn) enhancements** The following options have been added to the dbdsn utility:

  - **-dr** includes the DRIVER= parameter when you list the command that was used to create a data source. This allows you to recreate data sources so that they use a different version of the ODBC driver than the one included with the current version of the software.

  - **-f** displays the name of the system information file (typically *.odbc.ini*) being used.

  - **-ns** tells dbdsn not to search for the system information file (typically *.odbc.ini*), but to use the existing environment variables to determine where the file should be. This is useful when the file specified by one or more of the environment variables does not exist, and a ODBC data source is being created.

- **-pe** encrypts the password field in the data source.

See "Data Source utility (dbdsn)" [*SQL Anywhere Server - Database Administration*].

- **Histogram utility (dbhist) enhancements** Sheets within the Excel output file created by dbhist are now named to reflect the column name they apply to, instead of Sheet1, Sheet2, and so on. See "Histogram utility (dbhist)" [*SQL Anywhere Server - Database Administration*].

- **Information utility (dbinfo) enhancements** The -u option now includes information about materialized views. See "Information utility (dbinfo)" [*SQL Anywhere Server - Database Administration*].

- **Initialization utility (dbinit) enhancements** The Initialization utility (dbinit) now supports the following new options:

  - **-a** uses accent sensitivity for UCA string comparisons
  - **-af** uses French accent sensitivity rules for UCA string comparisons.
  - **-dba** changes the user ID and/or password of the default DBA database user in a new database.
  - **-dbs** specifies the initial size of the database file.
  - **-ze** specifies the character set encoding for the CHAR data type.
  - **-zn** specifies the collation sequence for the NCHAR data type.

  See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*].

- **Log Transfer Manager (LTM) enhancements** The Log Transfer Manager (LTM) utility, also known as the Replication Agent, now supports identifiers up to 128 bytes for table, column, procedure, function, and parameter names when using the Replication Agent with Replication Server 15.0 and Open Server/Open Client 15.0. In earlier versions of the software, identifiers were limited to 30 bytes.

  Timestamps in informational, warning, and error messages generated by dbltm now use the non-ambiguous ISO 8601 datetime format: `{I|W|E} yyyy-mm-dd hh:mm:ss message`.

  Note
  The Log Transfer Manager is unsupported as of SQL Anywhere version 12.

- **Ping utility (dbping) enhancements** You can use the Ping utility (dbping) to obtain information about the performance of embedded SQL connections and your network's performance by specifying the -s or -st options. These options report statistics about the performance between the computer running dbping and the computer running the database server. See "Testing embedded SQL connection performance" [*SQL Anywhere Server - Database Administration*].

  The -pd option now lets you specify the name of the database you want to obtain the property value from. See "Ping utility (dbping)" [*SQL Anywhere Server - Database Administration*].

- **Server Enumeration utility (dblocate) enhancements** The Server Enumeration utility (dblocate) now supports several new options to search for databases:

  - **-d** displays the server name and address, and a comma-separated list of all databases running on each server.

  - **-dn** displays the server name and address only if the server is running a database with the specified name.

  - **-dv** displays the server name and address, and lists all databases running on each server on a separate line.

  - **-p** displays servers using the specified TCP/IP port number.

  - **-s** displays servers with the specified name.

  - **-ss** displays server names that contain the specified substring.

  See "Server Enumeration utility (dblocate)" [*SQL Anywhere Server - Database Administration*].

- **Service utility (dbsvc) enhancements** The Service utility (dbsvc) supports the DBLTM service type, which allows you to manage services for the Log Transfer Manager, and the dblsn service type, which lets you manage services for the Listener utility.

  The Service utility also supports the -o option, which allows you to log output from the utility to a file. See "Service utility (dbsvc) for Windows" [*SQL Anywhere Server - Database Administration*] and "Service utility (dbsvc) for Linux" [*SQL Anywhere Server - Database Administration*].

- **New SQL Anywhere Broadcast Repeater utility (dbns10)** The SQL Anywhere Broadcast Repeater utility allows SQL Anywhere clients to find SQL Anywhere database servers on other subnets and through firewalls, where UDP broadcasts do not normally reach, without using the HOST parameter or LDAP. See "Broadcast Repeater utility (dbns12)" [*SQL Anywhere Server - Database Administration*].

- **New report submission utility (dbsupport)** The new Support utility (dbsupport) provides the ability to submit error reports and statistics, the ability to query for updates (availability of EBFs), and the ability to check if previously submitted problems have been fixed. See "Support utility (dbsupport)" [*SQL Anywhere Server - Database Administration*].

- **Unload utility (dbunload) enhancements** The following dbunload enhancements have been made:

  - unprocessed statements are logged when dbunload encounters a failure. See "Failed unloads" [*SQL Anywhere Server - Database Administration*].

  - support for binary data in unloaded tables.

  - many internal enhancements have been made to improve performance for unloading databases.

  The following new options have been added:

- **-dc** recalculates the values for all computed columns in the database.
- **-g** initializes materialized views during reload.
- **-k** creates an auxiliary table for tracing support. Specifying this option populates the sa_diagnostic_auxiliary_catalog table. This option is useful when creating a tracing database.
- **-nl** creates a *reload.sql* file that includes LOAD TABLE and INPUT statements for each table, but no data.

See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

- **Validation utility (dbvalid)** A new database validation option, -d, has been added. This option performs a database validation that includes checksum validation, checks for orphaned table pages and BLOBs, as well as structural checks. Data is not checked. See "Validation utility (dbvalid)" [*SQL Anywhere Server - Database Administration*].

Database options

The following database options have been added or have enhanced functionality:

- **ansi_substring database option** This option controls the behavior of the SUBSTRING function. By default, the behavior of the SUBSTRING function now corresponds to ANSI/ISO SQL/99 behavior. A negative or zero start offset is treated as if the string were padded on the left with non-characters, and gives an error if a negative length is provided.
- **collect_statistics_on_dml_updates database option** Controls the gathering of statistics during the execution of DML statements (INSERT, DELETE, and UPDATE). See "collect_statistics_on_dml_updates option" [*SQL Anywhere Server - Database Administration*].
- **default_dbspace option** This option allows you to specify the default dbspace where your tables are created. See "default_dbspace option" [*SQL Anywhere Server - Database Administration*].
- **http_session_timeout option** The http_session_timeout option provides variable session timeout control. The option setting is in units of minutes. By default the public setting is 30 minutes. Minimum is 1 minute and maximum is 525600 minutes (365 days). See "http_session_timeout option" [*SQL Anywhere Server - Database Administration*].
- **max_temp_space database option** You can specify the maximum amount of temporary space a connection can use with the max_temp_space option. See "max_temp_space option" [*SQL Anywhere Server - Database Administration*].
- **materialized_view_optimization database option** This option controls the optimizer's use of materialized views. See "materialized_view_optimization option" [*SQL Anywhere Server - Database Administration*].
- **oem_string database option** You can store information in the header page of a database file using the oem_string database option. This string can be accessed by applications and used for such

purposes as storing version information, or validating that the database file is intended for your application. See "oem_string option" [*SQL Anywhere Server - Database Administration*].

- **request_timeout database option** This option allows you to specify the maximum time a single request can run to help prevent connections from consuming a significant amount of server resources for a long period of time. See "request_timeout option" [*SQL Anywhere Server - Database Administration*].

- **synchronize_mirror_on_commit option** This option controls when database changes are assured to have been sent to a mirror server when running database mirroring in asynchronous or asyncfullpage mode. See "synchronize_mirror_on_commit option" [*SQL Anywhere Server - Database Administration*].

- **uuid_has_hyphens database option** This option controls the formatting of uniqueidentifier values when they are converted to strings.

- **verify_password_function database option** The verify_password_function database option allows you to specify a function that can be used to implement password rules. The function is called on a GRANT CONNECT statement. See "verify_password_function option" [*SQL Anywhere Server - Database Administration*].

- **New database options for Java support** The following database options have been added:

  - "java_location option" [*SQL Anywhere Server - Database Administration*]
  - java_main_userid option [database]
  - "java_vm_options option" [*SQL Anywhere Server - Database Administration*]

Database server options

In addition to any server options that have been documented in the New Features section, the following new server options have been added:

- **-cm server option** This server option allows you to specify the amount of address space allocated for Address Windowing Extensions (AWE) on Windows. See "-cm dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-dh server option** This server option makes a database undetectable when the Server Enumeration utility (dblocate) is run against the server. See "-dh dbeng12/dbsrv12 database option" [*SQL Anywhere Server - Database Administration*].

- **-dt server option** This server option allows you to specify the directory where temporary files are stored. This option cannot be specified for database servers using shared memory connections on Unix. See "-dt dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-gtc server option** This option lets you control the number of threads that can run concurrently on a CPU. See "-gtc dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-ot server option** When you specify this server option, the database server message log file is truncated before any messages are written to it. See "-ot dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-su server option** This option lets you specify the password for the DBA user when connecting to the utility database. You should use -su instead of the *util_db.ini* file. See "-su dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **-zp server option** When you specify this server option, the query optimization plan that was most recently used is stored for each connection. See "-zp dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Properties and Performance Monitor statistics

Several new connection, server, and database properties, as well as new Performance Monitor statistics, have been added to help you administer your database.

- **Connection properties** The following connection properties have been added in this release:
  - allow_snapshot_isolation
  - ansi_substring
  - ApproximateCPUTime
  - conn_auditing
  - default_dbspace
  - ExprCacheAbandons
  - ExprCacheDropsToReadOnly
  - ExprCacheEvicts
  - ExprCacheHits
  - ExprCacheInserts
  - ExprCacheLookups
  - ExprCache
  - GetData
  - HeapsCarver
  - HeapsLocked
  - HeapsQuery
  - HeapsRelocatable
  - HttpServiceName
  - http_session_timeout
  - java_location
  - java_main_userid
  - LastPlanText
  - LockCount
  - LockedCursorPages
  - LockTableOID
  - materialized_view_optimization
  - max_query_tasks
  - max_temp_space
  - MultiPageAllocs
  - NcharCharSet
  - oem_string
  - post_login_procedure
  - QueryHeapPages
  - ReqCountActive
  - ReqCountBlockContention
  - ReqCountBlockLock
  - ReqCountBlockIO
  - ReqCountUnscheduled
  - ReqTimeActive
  - ReqTimeBlockContention
  - ReqTimeBlockIO
  - ReqTimeBlockLock
  - ReqTimeUnscheduled
  - ReqStatus
  - RequestsReceived
  - RetryConnectionTimeout

- SessionCreateTime
- SessionID
- SessionLastTime
- SnapshotCount
- synchronize_mirror_on_commit
- tsql_outer_joins
- verify_password_function

For more information about connection properties, see "Connection properties" [*SQL Anywhere Server - Database Administration*].

- **Server properties** The following server properties have been added in this release:

  - CachePinned
  - CacheReadEng
  - CacheSizingStatistics
  - CarverHeapPages
  - ConsoleLogMaxSize
  - CollectStatistics
  - DebuggingInformation
  - DefaultNcharCollation
  - DiskReadEng
  - ExchangeTasks
  - FirstOption
  - FunctionMaxParms
  - FunctionMinParms
  - HeapsRelocatable
  - HeapsLocked
  - HeapsQuery
  - HeapsCarver
  - IsEccAvailable
  - IsRsaAvailable
  - LastConnectionProperty
  - LastDatabaseProperty
  - LastOption
  - LastServerProperty
  - MapPhysicalMemoryEng
  - MaxConnections
  - MultiProgrammingLevel
  - NumLogicalProcessors
  - NumLogicalProcessorsUsed
  - NumPhysicalProcessors
  - NumPhysicalProcessorsUsed
  - QueryHeapPages
  - RememberLastPlan
  - RemoteputWait
  - RequestFilterConn
  - RequestFilterDB
  - RequestLogMaxSize
  - RequestsReceived
  - ServerName
  - StartDBPermission

  For more information about these properties, see "Database server properties" [*SQL Anywhere Server - Database Administration*].

- **Database properties** The following database properties have been added in this release:

  - AccentSensitive
  - AlternateServerName
  - ArbiterState
  - AuditingTypes
  - CleanablePagesAdded
  - CleanablePagesCleaned
  - EncryptionScope
  - http_session_timeout
  - IOParallelism
  - JavaVM
  - LockCount
  - MirrorState
  - NcharCollation
  - NcharCharSet
  - NextScheduleTime
  - PartnerState
  - ReceivingTracingFrom
  - SendingTracingTo
  - SnapshotCount
  - SnapshotIsolationState
  - VersionStorePages
  - XPathCompiles

  For more information about these properties, see "Database properties" [*SQL Anywhere Server - Database Administration*].

- **Performance monitor statistics properties** The following Performance Monitor statistics have been added in this release:

  - Cache: Multi-Page Allocations
  - Cache: Panics
  - Cache: Scavenge Visited
  - Cache: Scavenges
  - Cache Pages: Allocated Structures
  - Cache Pages: File
  - Cache Pages: File Dirty
  - Cache Pages: Free
  - Comm: Requests Received
  - Heaps: Carver
  - Heaps: Query Processing
  - Heaps: Relocatable Locked
  - Heaps: Relocatable
  - Mem Pages: Carver
  - Mem Pages: Pinned Cursor
  - Mem Pages: Query Processing
  - Version Store Pages

For more information about these statistics, see "Performance Monitor statistics" [*SQL Anywhere Server - SQL Usage*].

System procedures and functions

Following are several new system procedures and functions, and new extensions to existing system procedures and functions.

- **Enhancements to all procedures and functions to support the DEFAULT clause** For procedures and user-defined functions, the value DEFAULT may be provided as an argument if the corresponding parameter was defined with a default value. When the procedure has several parameters and the ones being defaulted are not all at the end, it may be easier to specify DEFAULT in the argument list than to use named parameters. Also, named parameters are not permitted in function calls.

- **New system procedures** The following system procedures have been added:

  - **sa_clean_database system procedure** Runs the database cleaner for the time specified. See "sa_clean_database system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_column_stats system procedure** The sa_column_stats system procedure returns string-related statistics about the specified column(s). See "sa_column_stats system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_conn_list system procedure** The sa_conn_list system procedure returns a connection ID. See "sa_conn_list system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_conn_options system procedure** The sa_conn_options system procedure returns property information for connection properties that correspond to database options. See "sa_conn_options system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_db_list system procedure** The sa_db_list system procedure returns a database ID. See "sa_db_list system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_describe_query system procedure** The sa_describe_query system procedure returns one row per column and describes the domain of the result expression and its nullability. This procedure is equivalent to performing the EXPRTYPE function on each column. See "sa_describe_query system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_get_bits system procedure** The sa_get_bits system procedure decodes a bit string, returning one row for each bit in the bit string, indicating the value of the bit. See "sa_get_bits system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_make_object system procedure** You can now specify an events as an object type for the sa_make_object system procedure. See "sa_make_object system procedure" [*SQL Anywhere Server - SQL Reference*].

  - **sa_materialized_view_info system procedure** The sa_materialized_view_info system procedure returns information about a specified materialized view, such as its status and the owner

of the view. See "sa_materialized_view_info system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_refresh_materialized_views system procedure** The sa_refresh_materialized_views system procedure refreshes all materialized views in the database that are currently in an uninitialized state. See "sa_refresh_materialized_views system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_remove_tracing_data system procedure** This procedure permanently deletes all record of a given logging session from the diagnostic tracing tables. See "sa_remove_tracing_data system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_save_trace_data system procedure** This procedure saves data from temporary tracing tables to the base tables. See "sa_save_trace_data system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_set_tracing_level system procedure** Sets the level of tracing data to generate for the database being profiled. See "sa_set_tracing_level system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_snapshots system procedure** Returns a list of snapshots that are currently active for the database. See "sa_snapshots system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_split_list system procedure** Takes a string representing a list of values and returns a result set containing that list. See "sa_split_list system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_table_stats system procedure** Returns information about how many pages have been read from each table. See "sa_table_stats system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_transactions** Returns a list of transactions that are currently running against a database. See "sa_transactions system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_unload_cost_model and sa_load_cost_model system procedures** You can now unload the cost model from one database and load it into another database using the new system procedures sa_unload_cost_model and sa_load_cost_model, respectively. This eliminates repetitive, time-consuming recalibration activities when there is a large number of similar hardware installations. See "sa_unload_cost_model system procedure" [*SQL Anywhere Server - SQL Reference*] and "sa_load_cost_model system procedure" [*SQL Anywhere Server - SQL Reference*].

- **New functions** The following functions have been added:
  - **BIT_LENGTH function** Returns the number of bits stored in the array. See "BIT_LENGTH function [Bit array]" [*SQL Anywhere Server - SQL Reference*].
  - **BIT_SUBSTR function** Returns a sub-array of a bit array. See "BIT_SUBSTR function [Bit array]" [*SQL Anywhere Server - SQL Reference*].

- **BIT_AND function** Takes two bit arrays and returns a bitwise AND-ing of its arguments using the following logic: for each bit compared, if both bits are 1, return 1; otherwise, return 0. See "BIT_AND function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].
- **BIT_OR function** Takes two bit arrays and returns a bitwise OR-ing of its arguments using the following logic: for each bit compared, if either bit (or both) is 1, return 1; otherwise, return 0. See "BIT_OR function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].
- **BIT_XOR function** Takes two bit arrays and returns a bitwise exclusive OR-ing of its arguments using the following logic: for each bit compared, if just one bit (but not both) is 1, return 1; otherwise, return 0. See "BIT_XOR function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].
- **COUNT_SET_BITS function** Returns a count of the number of bits set to 1 (TRUE) in the array. See "COUNT_SET_BITS function [Bit array]" [*SQL Anywhere Server - SQL Reference*].
- **GET_BIT function** Returns the value (1 or 0) of a specified bit in a bit array. See "GET_BIT function [Bit array]" [*SQL Anywhere Server - SQL Reference*].
- **REVERSE function** This new function returns the reverse of a character expression. See "REVERSE function [String]" [*SQL Anywhere Server - SQL Reference*].
- **SET_BIT function** Sets the value of a specific bit in a bit array. See "SET_BIT function [Bit array]" [*SQL Anywhere Server - SQL Reference*].
- **SET_BITS function** Creates a bit array where specific bits, corresponding to values from a set of rows, are set to 1 (TRUE). See "SET_BITS function [Aggregate]" [*SQL Anywhere Server - SQL Reference*].
- **TRACED_PLAN function** Generates a graphical plan for a query using tracing data and information about optimizer conditions when the query was traced. See "TRACED_PLAN function [Miscellaneous]" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to various system procedures and functions** The following system procedures and functions have been enhanced as described:
  - **Enhancements to property functions** Property functions can now return LONG VARCHAR.

    See:

    - "CONNECTION_PROPERTY function [System]" [*SQL Anywhere Server - SQL Reference*]
    - "DB_PROPERTY function [System]" [*SQL Anywhere Server - SQL Reference*]
    - "PROPERTY function [System]" [*SQL Anywhere Server - SQL Reference*]
  - **Enhancements to DB_EXTENDED_PROPERTY function** You can now use the DB_EXTENDED_PROPERTY function with the NextScheduleTime database property to obtain the next scheduled execution time for an event. You can also use the function the return extended information about the CHAR character set. See "DB_EXTENDED_PROPERTY function [System]" [*SQL Anywhere Server - SQL Reference*].

- **New CONNECTION_EXTENDED_PROPERTY function** You can use the CONNECTION_EXTENDED_PROPERTY function to find out extended information for certain connection parameters. See "CONNECTION_EXTENDED_PROPERTY function [String]" [*SQL Anywhere Server - SQL Reference*].

- **sa_procedure_profile system procedure** The output from sa_procedure_profile system procedure can now be saved to a file, has new syntax, requires fewer parameters, and has new uses. See "sa_procedure_profile system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_procedure_profile_summary system procedure** The sa_procedure_profile_summary system procedure now supports saving its output to a file, has new syntax, accepts fewer parameters, and has new uses. See "sa_procedure_profile_summary system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_server_option system procedure** The sa_server_option system procedure lets you change settings for the database server while it is still running. You can now change the following settings:

  - **CacheSizingStatistics property** Display cache information in the database server messages window whenever the cache size changes.

  - **CollectStatistics property** Collect Performance Monitor statistics for the database server.

  - **ConsoleLogFile property** Specify the name of the output file where database server messages window information is recorded.

  - **ConsoleLogMaxSize property** Specify the maximum size of the output file used to record database server messages window information.

  - **DebuggingInformation property** Display diagnostic communication messages and other messages for troubleshooting purposes.

  - **IdleTimeout server option** Disconnect TCP/IP or SPX connections that have not submitted a request for the specified number of minutes.

  - **ProfileFilterConn property** Capture profiling information for a specific connection ID, without preventing other connections from using the database.

  - **RequestFilterDB property** You can use the sa_server_option system procedure to filter connections to a single database for request logging.

  - **RequestLogging property** The request log can now record blocking and unblocking events, plan information, procedures, and triggers.

  - **RequestTiming property** Turning on request timing instructs the database server to maintain timing information for each request.

  For more information, see "sa_server_option system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to xp_startsmtp system procedure** The xp_startsmtp system procedure supports three new parameters: smtp_user_name, smtp_auth_username, and smtp_auth_password. See "xp_startsmtp system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to xp_sendmail system procedure** The xp_sendmail system procedure now supports attachments when sending mail using SMTP, using the new include_file parameter. In addition, xp_sendmail supports MIME content when using SMTP mail, using the new content_type parameter. See "xp_sendmail system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_conn_info system procedure now returns several new property values** The sa_conn_info system procedure now returns the following additional properties: ClientPort, ServerPort, and LockTable. The procedure no longer returns the LastIdle property, and the UncmtOps value has been renamed to UncommitOps. See "sa_conn_info system procedure" [*SQL Anywhere Server - SQL Reference*].

- **sa_performance_diagnostics returns more information** The sa_performance_diagnostics system procedure now returns the LockCount and SnapshotCount when you use snapshot isolation. See "sa_performance_diagnostics system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to the HASH function** The HASH function now accepts the following new algorithms: SHA256, SHA1_FIPS, and SHA256_FIPS. The FIPS related algorithms are only for use on systems that use FIPS-certified software. See "HASH function [String]" [*SQL Anywhere Server - SQL Reference*].

- **COMPRESS and DECOMPRESS functions support new algorithm** The gzip algorithm is now available to compress and decompress a string in a function. See "COMPRESS function [String]" [*SQL Anywhere Server - SQL Reference*] and "DECOMPRESS function [String]" [*SQL Anywhere Server - SQL Reference*].

SQL statements

Following are several new SQL statements, and new extensions to existing SQL statement syntax. These new features are in addition to statement changes listed in the previous feature sections of this document.

- **SQL statements to support materialized views** The following SQL statements have been added, or have had their syntax and functionality extended, to support materialized views:

  - "ALTER MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*]
  - "COMMENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE INDEX statement" [*SQL Anywhere Server - SQL Reference*]
  - "CREATE MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*]
  - "DROP MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*]
  - "REFRESH MATERIALIZED VIEW statement" [*SQL Anywhere Server - SQL Reference*]
  - "VALIDATE statement" [*SQL Anywhere Server - SQL Reference*]

A new OPTION clause in the SELECT statement can be used to override the materialized_view_optimization database option. See "SELECT statement" [*SQL Anywhere Server - SQL Reference*].

- **New SQL statements to support diagnostic tracing and application profiling** The new SQL statements to support the Application Profiling feature are listed below:
  - "ATTACH TRACING statement" [*SQL Anywhere Server - SQL Reference*]
  - "DETACH TRACING statement" [*SQL Anywhere Server - SQL Reference*]
  - "REFRESH TRACING LEVEL statement" [*SQL Anywhere Server - SQL Reference*]
- **New VALIDATE DATABASE statement** You can now validate the database using the VALIDATE DATABASE statement. See "VALIDATE statement" [*SQL Anywhere Server - SQL Reference*].
- **New VALIDATE MATERIALIZED VIEW statement** You can now validate materialized views using the VALIDATE MATERIALIZED VIEW statement. See "VALIDATE statement" [*SQL Anywhere Server - SQL Reference*].
- **New ALTER STATISTICS statement** You can now control whether column statistics are automatically updated using the ALTER STATISTICS statement. You can still force an update of statistics on columns where automatic updating has been disabled, using an explicit CREATE STATISTICS or DROP STATISTICS statement. See "ALTER STATISTICS statement" [*SQL Anywhere Server - SQL Reference*].
- **ALTER INDEX statement enhancement** You can now rebuild an index using the REBUILD clause of the ALTER INDEX statement. See "ALTER INDEX statement" [*SQL Anywhere Server - SQL Reference*].
- **ALTER TABLE and CREATE TABLE statement enhancements** You now have finer control over what constitutes a match between a foreign key in a referencing table, and the primary key in the referenced table using the MATCH clause. You are also able to declare a foreign key as unique, thereby eliminating the need to declare uniqueness separately. See "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*] and "ALTER TABLE statement" [*SQL Anywhere Server - SQL Reference*].
- **New CALIBRATE PARALLEL READ clause for the ALTER DATABASE statement** Use the new CALIBRATE PARALLEL READ clause of the ALTER DATABASE statement to detect hardware capable of parallel input and output. You can retrieve the calibration result for a dbspace by querying the new IOParallelism extended database property using the DB_EXTENDED_PROPERTY function. See "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*], and "Database properties" [*SQL Anywhere Server - Database Administration*].
- **New PRIMARY KEY ON clause for COMMENT statement** You can now create remarks for primary keys using the PRIMARY KEY ON clause of the COMMENT statement. See "COMMENT statement" [*SQL Anywhere Server - SQL Reference*].
- **CREATE ENCRYPTED FILE statement enhancement to change encryption keys** Using extensions to the CREATE ENCRYPTED FILE statement, you can now change the encryption key used to encrypt a database, transaction log, or dbspace without unloading and reloading the database.

If the database is not encrypted, but table encryption is enabled, you can use the CREATE ENCRYPTED FILE statement to change the key used for table encryption. See "CREATE ENCRYPTED FILE statement" [*SQL Anywhere Server - SQL Reference*].

- **CREATE DATABASE statement enhancements** Three new clauses, ENCODING, NCHAR COLLATION, and ACCENT, have been added for improved handing of character sets. Also, a DATABASE SIZE clause has been added so you can specify the initial size of a database. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **SELECT statement enhancements** The FOR UPDATE clause, used in updating rows through a cursor, has been extended to allow column lists to restrict which columns can be modified using a subsequent positioned UPDATE statement. See "SELECT statement" [*SQL Anywhere Server - SQL Reference*].

  The FROM clause of a SELECT statement has been extended to support the READPAST table hint, which directs the database server to ignore locked rows, and the UPDLOCK table hint, which behaves similarly to XLOCK. See "FROM clause" [*SQL Anywhere Server - SQL Reference*].

  The SELECT statement has been extended to support an OPTION clause to control aspects of query optimization for that particular statement. The OPTION clause includes syntax for controlling the matching of materialized views via the MATERIALIZED VIEW OPTIMIZATION clause for this specific SELECT statement. A second clause, FORCE OPTIMIZATION, directs the database server to perform optimization on a query, even if the query qualifies for bypassing cost-based optimization. See "SELECT statement" [*SQL Anywhere Server - SQL Reference*].

- **LOAD TABLE and UNLOAD TABLE statement enhancements** The STRIP clause for the LOAD TABLE statement now accepts options that allow you to control whether leading blanks are stripped from unquoted values before they are inserted. Additional STRIP options let you fine tune how the data is stripped.

  The LOAD TABLE statement has also been extended to support the COMMENTS INTRODUCED BY option. This option allows you to specify the string used to identify comments in the input data. Any lines in the input that begin with the specified string are ignored during the load operation.

  Both the LOAD TABLE and UNLOAD TABLE statements have been extended to support the following options:

  - **ENCODING option** Used to specify the encoding to use when loading or unloading data.
  - **ROW DELIMITED BY option** Used to specify the string that indicates the end of an input record when bulk loading or unloading data.
  - **QUOTE option** Similar to the QUOTE option for the OUTPUT statement in Interactive SQL. See "OUTPUT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

  See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*] and "UNLOAD statement" [*SQL Anywhere Server - SQL Reference*].

- **VALIDATE INDEX statement enhancements** The syntax for VALIDATE INDEX has been enhanced to support index specifications. See "VALIDATE statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the ALTER INDEX statement to rename primary keys** You can now rename primary keys using the ALTER INDEX statement. See "ALTER INDEX statement" [*SQL Anywhere Server - SQL Reference*].

- **New CONTINUE statement** Use this statement to restart a loop. Statements in the loop following the CONTINUE statement are skipped. See "CONTINUE statement" [*SQL Anywhere Server - SQL Reference*].

- **New BREAK statement [T-SQL]** Use this statement to leave a compound statement or loop. See "BREAK statement [T-SQL]" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to the INSERT statement control updating default values during an INSERT** You can control whether default values are updated during an INSERT when a row already exists using the DEFAULTS ON | OFF clause. This new capability does not extend to the following default fields: DEFAULT TIMESTAMP, DEFAULT UTC TIMESTAMP, and DEFAULT LAST USER; these fields are always updated. See "INSERT statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to the DELETE statement to support an ORDER BY clause** The DELETE statement now supports the ORDER BY clause, which allows you to specify the order in which rows are deleted from the database. See "DELETE statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to the START DATABASE statement** The START DATABASE statement now returns a wider range of error messages when the statement fails to indicate the reason why the database failed to start. As well, the START DATABASE clauses can now be specified in any order. See "START DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancement to the MESSAGE statement to support logging only to event or system log** In addition to being able to turn on or off logging, you can also specify whether to log only to the Event or System log. Syntax for the MESSAGE statement has been extended to allow the optional clause [EVENT | SYSTEM] in within the TO LOG clause. For example, TO EVENT LOG results in logging only to the Event log. See "MESSAGE statement" [*SQL Anywhere Server - SQL Reference*].

- **FOR OLAP WORKLOAD option** The syntax for the CREATE INDEX, CREATE TABLE, and ALTER TABLE statements has been extended to support a FOR OLAP WORKLOAD option in the foreign key definition. This option instructs the database server to perform certain optimizations and gather statistics on the key to improve OLAP performance. See "CREATE INDEX statement" [*SQL Anywhere Server - SQL Reference*], "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*], "ALTER TABLE statement" [*SQL Anywhere Server - SQL Reference*], and "ClusteredHashGroupBy algorithm (GrByHClust)" [*SQL Anywhere Server - SQL Usage*].

- **Support for temporary stored procedures** You can now create temporary stored procedures using an extension to the CREATE PROCEDURE statement. Temporary stored procedures are visible only by the connection that created them, and are automatically dropped when the connection is

dropped. See "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*].

- **Support for local temporary tables** You can now create local temporary tables using the CREATE LOCAL TEMPORARY TABLE statement. Local temporary tables created this way are dropped when the connection closes. See "CREATE LOCAL TEMPORARY TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **Enhancements to temporary tables** You can now create global temporary tables whose data can be shared by all connections to a database, using the SHARE BY ALL clause of the CREATE GLOBAL TEMPORARY TABLE statement. See "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*].

Data types

- **Support for character-length semantics for CHAR and VARCHAR data types** When you specify a CHAR or VARCHAR column, you can now use character-length semantics. Character-length semantics allow you to express the length in characters, instead of bytes. See "CHAR data type" [*SQL Anywhere Server - SQL Reference*] and "VARCHAR data type" [*SQL Anywhere Server - SQL Reference*].

- **Support for bit array data types** SQL Anywhere now supports the VARBIT and LONG VARBIT data types. These data types are used to store bit arrays. See "Bit array data types" [*SQL Anywhere Server - SQL Reference*].

  The following functions have been added for use with bit array data types:

  - "BIT_LENGTH function [Bit array]" [*SQL Anywhere Server - SQL Reference*]
  - "BIT_SUBSTR function [Bit array]" [*SQL Anywhere Server - SQL Reference*]
  - "BIT_AND function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]
  - "BIT_OR function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]
  - "BIT_XOR function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]
  - "COUNT_SET_BITS function [Bit array]" [*SQL Anywhere Server - SQL Reference*]
  - "GET_BIT function [Bit array]" [*SQL Anywhere Server - SQL Reference*]
  - "SET_BIT function [Bit array]" [*SQL Anywhere Server - SQL Reference*]
  - "SET_BITS function [Aggregate]" [*SQL Anywhere Server - SQL Reference*]

Programming interfaces

- **ADO.NET 2.0 support** The ADO.NET driver has been updated to support version 2.0 of the .NET framework. Several new classes and methods have been added as part of this support. See "Namespace" [*SQL Anywhere Server - Programming*].

- **SQL Anywhere Explorer** The SQL Anywhere Explorer lets you connect to SQL Anywhere databases from within Visual Studio .NET. In addition, you can open Sybase Central and Interactive SQL directly from Visual Studio .NET.

- **iAnywhere JDBC driver supports JDBC 3.0** The iAnywhere JDBC driver now supports JDBC 3.0 calls. The iAnywhere JDBC driver no longer supports JDBC 2.0. Both the ianywhere.ml.jdbcodbc.IDriver and ianywhere.ml.jdbcodbc.jdbc3.IDriver classes are still supported to allow existing applications to continue running without modification, but, both drivers are now identical and implement JDBC 3.0 only. You can no longer use JRE versions earlier than 1.4 with the iAnywhere JDBC driver. See "JDBC support" [*SQL Anywhere Server - Programming*].

- **iAnywhere JDBC driver supports the SQL Server Native Client ODBC driver** The iAnywhere JDBC driver now checks if the ODBC driver is the Microsoft SQL Server Native Client ODBC driver and appropriately sets the default result set type and other attributes.

- **Support for PreparedStatement.addBatch method** The iAnywhere JDBC driver now supports the PreparedStatement.addBatch method. This method is supported for batched (or wide) inserts.

- **Support for SQL_GUID added to ODBC driver** Support for UNIQUEIDENTIFIER columns has now been added to the SQL Anywhere ODBC driver. A UNIQUEIDENTIFIER column can now be typed as SQL_GUID.

- **Support for GUID escape sequences added to ODBC driver** Support for GUID escape sequences has been added to the SQL Anywhere ODBC driver. GUID escape sequences may be used in SQL statements prepared and executed through ODBC. A GUID escape sequence has the form {guid 'nnnnnnnn-nnnn-nnnn-nnnn-nnnnnnnnnnnn'}.

- **ODBC message callbacks are now per-connection** ODBC has supported message callbacks since Adaptive Server Anywhere version 9.0.0, but messages for all connections came to a single callback function. As of version 9.0.2, when you designate a message callback function, it applies only to a single connection. This is consistent with how DBLIB works. All messages now funnel through a single function in the ODBC driver, which filters the messages by connection, and only calls the connection's callback function for those connections that have one.

- **New functions added to the SQL Anywhere PHP module** The following new functions have been added to the SQL Anywhere PHP module:

  - sqlanywhere_execute
  - sqlanywhere_error
  - sqlanywhere_errorcode
  - sqlanywhere_insert_id

  In addition, two new options have been added to the sqlanywhere_set_option function: verbose_errors and row_counts. See "SQL Anywhere PHP API reference" [*SQL Anywhere Server - Programming*].

- **Enhancements to db_locate_servers_ex function** The db_locate_servers_ex function supports two new flags: DB_LOOKUP_FLAG_ADDRESS_INCLUDES_PORT, which returns the TCP/IP port number in the a_server_address structure passed to the callback function, and DB_LOOKUP_FLAG_DATABASES, which indicates that the callback function is called once for each database or database server that is found. See "db_locate_servers_ex function" [*SQL Anywhere Server - Programming*].

- **Perl DBD::ASAny driver for the Perl DBI module renamed** The Perl driver has been renamed from DBD::ASAny to DBD::SQLAnywhere. Perl scripts that use SQL Anywhere must be changed to

use the new driver name. The cursor attribute ASATYPE, which returns native SQL Anywhere types has not changed, and neither have the type names (ASA_STRING, ASA_FIXCHAR, ASA_LONGVARCHAR, and so on). See "Perl DBI support" [*SQL Anywhere Server - Programming*].

- **SQL preprocessor (sqlpp) -o option values** The sqlpp -o option now accepts WINDOWS rather than WINNT for Microsoft Windows. As well, you can specify UNIX64 for supported 64-bit Unix operating systems. See "SQL preprocessor" [*SQL Anywhere Server - Programming*].

New ODBC Driver Manager and ODBC driver enhancements

- **ODBC driver manager enhancements** The ODBC Driver Manager now supports: all ODBC 3.x calls, wide CHAR entry points, tracing of connections. In addition, the ODBC Driver Manager is now able to switch between a non-threaded or threaded SQL Anywhere driver.

- **ODBC Driver Manager can now be used by both threaded and non-threaded applications** The ODBC Driver Manager can now be used by both threaded and non-threaded applications.

Deployment

- **Deployment wizard** The Deployment wizard has been added for creating deployments of SQL Anywhere for Windows. The Deployment wizard can be used to create both Microsoft Windows Installer package files and Microsoft Windows Installer Merge Module files. The InstallShield merge modules and templates provided with previous versions of SQL Anywhere are no longer supplied. Instead, use the Deployment wizard to create SQL Anywhere deployments. See "Using the Deployment Wizard" [*SQL Anywhere Server - Programming*].

Windows CE enhancements

- **Dynamic cache sizing supported on Windows CE** Windows CE now supports dynamic cache sizing. Like Windows and Unix, on Windows CE the size of the database server cache increases and decreases depending on the load on the database server and the other demands on system memory. This feature removes the need for choosing an explicit cache size under in many circumstances, and can also boost performance.

- **Creating proxy ports for Windows CE** In previous releases of the software, you had to modify entries in the registry to configure ActiveSync to use a proxy port for connecting to a database on Windows CE devices. The Connect window for Interactive SQL, Sybase Central, and the SQL Anywhere Console utility now includes a Setup Windows CE Proxy Port tool that allows you to create proxy ports for connecting to databases on Windows CE devices without editing the registry.

- **Ability to rebuild databases on Windows CE** You can now rebuild your database on Windows CE using a two step process whereby you unload your database to file and then reload it into a new database using the dbunload utility. See "Rebuilding databases on Windows Mobile" [*SQL Anywhere Server - Database Administration*].

- **dbrunsql utility** The Script Execution utility (dbrunsql) allows you to type SQL commands or run command files on Windows CE. See "Script Execution utility (dbrunsql)" [*SQL Anywhere Server - Database Administration*].

- **Signed .cab files for Windows Mobile 5** The SQL Anywhere installation includes pre-built *.cab* files that are signed by Verisign. When you use these *.cab* files, you do not receive an unknown publisher warning.

Unix/Linux enhancements

- **New ODBC driver manager that can be used on Unix platforms** The libdbodm10 shared object can now be used on Unix platforms as the ODBC Driver Manager. Applications using the iAnywhere ODBC Driver Manager must restrict their ODBC reliance to version 3.0 and later. See "Using the SQL Anywhere ODBC driver manager on Unix" [*SQL Anywhere Server - Programming*].

- **-uf server option** The -uf option allows you to specify the action taken by the database server when a fatal error occurs on Unix. See "-uf dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Service utility now supported on Linux** The Service utility is now available for use on Linux to create, delete, list, start, and stop SQL Anywhere services.

- **Additional samples supported on Unix** The following SQL Anywhere samples are now supported on Unix:

  - **DBTools** This sample is a database tools application that illustrates how to call and compile the database tools library by making a backup of the SQL Anywhere sample database.

  - **DiskFull** This sample illustrates a sample Disk-Full Callback DLL.

  - **HTTP** The samples in the HTTP directory demonstrate a variety of web service functionality, including how to use web services to set and retrieve client-side cookies within a SQL procedure, how to handle binary data within an HTTP web service procedure, how to use forms and HTML tables to display a simple calendar, how to use xp_read_file to retrieve images from the local disk and return them as the response to an HTTP request, and how to create, use, and delete HTTP sessions.

  - **oemString** This sample shows you how to determine if a database file is set up for an OEM's software.

  - **PerformanceFetch** This sample shows you how to use fetchtest to test fetch rates for an arbitrary query.

  - **PerformanceInsert** This sample shows you how to use instest to test insert rates into a table.

- **Support for fibers on Linux (thread affinity) and improved concurrent processing** The SQL Anywhere for Linux database server introduces a new co-routine processing model similar to that of Windows fibers. This processing model enables the server more control over context switching between tasks/routines providing better affinity to database operations.

- **Direct I/O support for Linux** SQL Anywhere for Linux now supports the Linux 2.6 O_DIRECT feature, which can improve I/O performance because the file system does not cache the I/O.

- **Asynchronous I/O for Linux** SQL Anywhere for Linux now supports the Linux AIO feature that enables even a single application thread to overlap I/O operations with other processing improving IO performance.

- **Linux desktop GUI** SQL Anywhere for Linux now offers an optional desktop GUI for the personal database server, network database server, MobiLink server, MobiLink synchronization client, and SQL Remote. This GUI can be invoked with the -ui option if you have GTK libraries installed.

- **Linux desktop icons** SQL Anywhere for Linux now offers optionally installed icons for the Linux Desktop to improve the usability of starting and managing of database servers, Sybase Central, Interactive SQL, the SQL Anywhere Console utility, and the MobiLink Monitor for users running the Linux Desktop.

- **64-bit client support for IBM AIX, HPUX, and Sun Solaris** SQL Anywhere client libraries are now available for the 64-bit memory model enabling you to take advantage of larger memory within your applications on IBM AIX, HP-UX, and Sun Solaris.

Web services

- **Web server is HTTP 1.1 compliant** For HTTP 1.1 compliance, the web server now accepts the following items:

  - pipelining of HTTP requests, enabling multiple HTTP requests such as GET and HEAD to be processed simultaneously
  - absolute URIs (previously only relative URIs were supported)
  - 100-continue request-header field, enabling a client to determine if the server would accept a request (based on the request headers) before the client sends the entire request body.
  - quality values in the Accept-Charset request-header field (these values were previously ignored)

- **HTTP client is HTTP 1.1 compliant** The following HTTP-related enhancements are now implemented:

  - **Support for HTTP string memory pooling** HTTP strings are no longer stored in contiguous memory. The cache is used as backend storage.

  - **Client chunk mode** An HTTP client can now send a POST request using HTTP chunked mode.

  - **HTTP sessions** HTTP connections can create an HTTP session to maintain state between HTTP requests.

- **HTTP server supports keep-alive** The database server now supports the keep-alive option when requested by HTTP clients. Instead of closing a connection after each request, an HTTP connection can be kept open after each request and response, so that multiple requests can be executed on the same connection.

The KeepaliveTimeout protocol option has also been added to support this feature. See "KeepaliveTimeout (KTO) protocol option" [*SQL Anywhere Server - Database Administration*].

- **New HttpServiceName connection property** A new connection property, HttpServiceName, has been added to enable a web application to determine its service name origin. The property is useful for error reporting and control flow. See "Connection properties" [*SQL Anywhere Server - Database Administration*].

- **sa_set_http_option enhancements** You can now use the sa_set_http_option system procedure to control the character set used in the HTTP response based on the request Accept-Charset request-header field. See "sa_set_http_option system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Support for data typing for SOAP services** The CREATE SERVICE and ALTER SERVICE statements have been extended to support a new DATATYPE clause. This clause is for use with SOAP services only, and controls whether data typing is supported for input parameters and output responses. See "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*] and "ALTER SERVICE statement" [*SQL Anywhere Server - SQL Reference*].

- **sa_set_soap_header system procedure** Use the sa_set_soap_header system procedure to set the response headers for SOAP services. See "sa_set_soap_header system procedure" [*SQL Anywhere Server - SQL Reference*].

- **SOAP_HEADER and NEXT_SOAP_HEADER functions** Use the SOAP_HEADER function to get request headers for SOAP services. See "SOAP_HEADER function [SOAP]" [*SQL Anywhere Server - SQL Reference*].

  Use the NEXT_SOAP_HEADER function to get the next header entry in a SOAP header. See "NEXT_SOAP_HEADER function [SOAP]" [*SQL Anywhere Server - SQL Reference*].

- **HEADER clause in CREATE PROCEDURE, ALTER PROCEDURE, CREATE FUNCTION, and ALTER FUNCTION statements** A new HEADER clause has been added to these statements, for use when creating HTTP web service client procedures and functions. This clause allows you to add or modify HTTP request header entries.

  See "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*], "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*], and "Accessing client-supplied HTTP variables and headers" [*SQL Anywhere Server - Programming*].

- **SOAPHEADER clause in CREATE PROCEDURE, ALTER PROCEDURE, CREATE FUNCTION, and ALTER FUNCTION statements** A new SOAPHEADER clause has been added to these statements, for use when creating SOAP web service client procedures and functions. This clause allows you to specify the SOAP header entries sent, and the SOAP header data received, using IN (IN/OUT) substitution parameters.

  See "CREATE PROCEDURE statement (web clients)" [*SQL Anywhere Server - SQL Reference*], "CREATE FUNCTION statement (web clients)" [*SQL Anywhere Server - SQL Reference*], and "Tutorial: Using SQL Anywhere to access a SOAP/DISH service" [*SQL Anywhere Server - Programming*].

Miscellaneous

- **Indexing enhancements** The following enhancements have been made to indexing in this release:

  - **Support for index sharing** When you create a primary key, secondary key, foreign key, or unique constraint, you now create a logical index that points to a physical index (an actual indexing structure on disk). The database server automatically determines whether a new physical index is required to satisfy the logical index. This model allows the sharing of physical indexes and prevents the creation and maintenance of duplicate physical indexes, which wastes disk space. See "Index sharing using logical indexes" [*SQL Anywhere Server - SQL Usage*].

  - **Improved storage of index information** There are several improvements to how index information is organized in the database. For example, the list of all indexes, including primary and foreign key indexes, is now stored in a single system table, ISYSIDX.

    Three new system tables, ISYSPHYSIDX, ISYSIDXCOL, and ISYSFKEY provide additional information about the indexes listed in ISYSIDX. See:

    - "Index information in the catalog" [*SQL Anywhere Server - SQL Usage*]
    - "SYSIDX system view" [*SQL Anywhere Server - SQL Reference*]
    - "SYSPHYSIDX system view" [*SQL Anywhere Server - SQL Reference*]
    - "SYSIDXCOL system view" [*SQL Anywhere Server - SQL Reference*]
    - "SYSFKEY system view" [*SQL Anywhere Server - SQL Reference*]

  - **Index Consultant enhancements** The Index Consultant has been enhanced to improve recommendations for clustered indexes, database and server states in the workload, and complete workload statistics reporting. It has been integrated into the Application Profiling tool.

  - **Improved control over how indexes are created** When an application creates a referential integrity constraint (primary key, foreign key, or unique constraint), the database server enforces the constraint by implicitly creating an index on the columns that make up the key of the constraint. The database server now allows you to specify how that index is created. You can specify the order of columns in the constraint key and the sequencing of values (ascending or descending) for each column in the index. In addition, there is no requirement that the order and sequencing of columns in a foreign key match the corresponding primary key or unique constraint.

    Additional improvements include:

    - The primary key order can now be changed without having to reorder the columns in the table.
    - The sequencing of columns in all constraint indexes can be specified to match application requirements.
    - Foreign key indexes can now be tailored to match the application requirements for the foreign key table without being tied to the primary table design.
    - Foreign keys can now have unique constraints.

- **New outer join elimination rewrite optimization** Outer joins are eliminated from the query before execution if the resulting query is semantically equivalent to the original query. See "Semantic query transformations" [*SQL Anywhere Server - SQL Usage*].

- **Date format strings now use character-length semantics** Date format strings now use character-length semantics to control the amount of text substituted for format specifiers; previously, byte-length semantics were used. For example, when formatting dates using strings, MMM used to imply the use of 3 bytes to store the month, now it implies 3 characters.

- **Directory access servers** You can now create a remote server that accesses the directory structure of the computer running the database server by creating a directory access server. See "Using directory access servers" [*SQL Anywhere Server - SQL Usage*].

- **Norwegian collation** 1252NOR has been added to support Norwegian. On Norwegian Windows systems, the database server chooses 1252NOR as the default collation for a new database if no collation is specified. See "Supported and alternate collations" [*SQL Anywhere Server - Database Administration*].

- **UTF8BIN collation** The UTF8BIN collation has been added to offer improved sorting of binary data. This new collation replaces the UTF8 collation, which is now deprecated. See "Supported and alternate collations" [*SQL Anywhere Server - Database Administration*].

- **Database server messages window enhancements** The following enhancements have been made to the database server messages window:

  - **New right-click choices for the window title bar** On all supported Windows platforms (except Windows CE), when you right-click the title bar of the database server messages window you can now choose About or Clear Message Area. Choosing About displays information about the database server, while choosing Clear Messages Area erases all of the messages in the database server messages window. Replicas of this window (the database server message log file, the Sybase Central Server Messages and Executed SQL pane, and the SQL Anywhere Console utility) are not affected by the clearing action.

  - **Environment variables used by database server can be logged to the database server messages window** The -ze server option displays a list of the environment variables for a database server in the database server messages window. This feature is not available on NetWare or Windows CE. See "-ze dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

  - **Controlling window minimization on startup** By default, the database server messages window minimizes once the database server starts. You can specify the -qn option if you do not want the database server messages window to minimize once the database server is started. See "-qn dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Ability to track when a table was last updated** The database server now keeps track of the last time a table was updated. This is achieved using the new last_modified_at column in the SYSTAB system view. See "SYSTAB system view" [*SQL Anywhere Server - SQL Reference*].

- **SNMP traps when changing to another server during mirroring** The SQL Anywhere SNMP Extension Agent now sends a trap when it is connected to a server involved in mirroring, the connection drops, and a new connection is reestablished but to a *different* server.

  This trap indicates that the original server went down, and the server that was acting as the mirror became the primary. See "Using traps" [*SQL Anywhere Server - Database Administration*].

- **Changes to request logging** The request log is now stored in a comma-delimited text format, reducing it to roughly one third of its original size. Also, where possible, instead of a normal time entry, times are now recorded as either an equal sign (=), which means the same time as the previous entry in the log, or *+nnn*, where *nnn* is the number of milliseconds after the previous entry in the log. Additional information is now also recorded. For example, for queries, the isolation level, number of rows fetched, and cursor type are now recorded. For INSERT, UPDATE, and DELETE statements, the number of rows affected and number of triggers fired are now recorded. See "Request logging" [*SQL Anywhere Server - SQL Usage*].

  The sa_get_request_times system procedure supports only the new request log format. However, the tracetime Perl script, *tracetime.pl*, processes both old and new request log formats. The tracetime script also performs faster on logs of the new format, noticeably so on large request logs.

- **ODBC driver enhancements** SQL Anywhere is using new drivers for remote data access when connecting to Adaptive Server Enterprise and DB2 databases. See: "Changes to ODBC drivers used by MobiLink, QAnywhere, and remote data access" on page 301.

- **SQLANYSAMP10 environment variable** The SQLANYSAMP10 environment variable specifies the location of the directory containing the SQL Anywhere 10 samples, including the *demo.db* and the *custdb.db* sample databases. See "SQLANYSAMP12 environment variable" [*SQL Anywhere Server - Database Administration*].

Version support

This section includes changes to the version numbers of supported and unsupported third-party components.

- **Support for version 8 and earlier databases** When using Sybase Central, Interactive SQL, or the SQL Anywhere Console utility, databases created with versions of the software older than Adaptive Server Anywhere 8.0.0 are no longer supported. This includes databases that were created with older software and then upgraded using newer software. Attempting to load such databases will result in an error on database startup. You can connect to these databases from Sybase Central to unload them into a new version 10 database. See "Upgrading SQL Anywhere" on page 304.

- **jConnect version 5.5 and 6.0.5 supported** SQL Anywhere now supports jConnect versions 5.5 and 6.0.5 (version 5.5 was supported in 9.0.2) for connecting to the database server. jConnect is available as a separate download from http://www.sybase.com/products/informationmanagement/softwaredeveloperkit/jconnect. Consult the jConnect documentation for information about its supported features.

- **Processor requirements on Intel x86 architectures** On Intel x86 architectures, SQL Anywhere supports only Pentium-class and newer processors, and will not start on older processors, such as the 80386 or 80486.

Behavior changes

Following is a list of changes to SQL Anywhere databases introduced in version 10.0.0, grouped by category.

Miscellaneous

- **Adaptive Server Anywhere renamed** In version 10.0.0, Adaptive Server Anywhere has been renamed SQL Anywhere.

- **Upgrade changes** The Upgrade Database wizard, the Upgrade utility (dbupgrad), and ALTER DATABASE UPGRADE statement cannot be used to upgrade version 9.0.2 and earlier databases to version 10. To upgrade older databases to version 10, you must rebuild the database by performing an unload and reload. See "Upgrading SQL Anywhere" on page 304.

- **Password changes** In newly-created databases, all passwords are case sensitive, regardless of the case-sensitivity of the database. The default DBA password for new databases is **sql**.

  When you rebuild an existing database, the case sensitivity of the password is determined as follows:

  - If the password was originally entered in a case-insensitive database, the password remains case-insensitive.
  - If the password was originally entered in a case-sensitive database, uppercase and mixed case passwords remain case sensitive. However, if the password was entered in all lowercase, then the password becomes *case-insensitive*.
  - Changes to existing passwords and new passwords are case sensitive.

  The database server now uses SHA256 to hash passwords. The old (proprietary) hashing algorithm is still supported for passwords reloaded from old databases, but all new passwords will use SHA256.

  Passwords are now stored in UTF-8, so they continue to work if the database is reloaded into a database with a different character set.

  In previous releases when connecting from embedded SQL, it was possible to connect to a database with DBA permission and then successfully make a second connection to the same database for any user without specifying the password. Now the password must be specified on every connection.

- **Blank padding changes** In previous releases of SQL Anywhere, the semantics of string comparisons with blank-padded databases was as if the two strings being compared were padded with an infinite number of blanks. In this version 10, these semantics have been changed so that the comparison is performed by ignoring trailing blanks in each string.

For equal (=) and not equal (<>) comparisons, there is no change in semantics; the two techniques (blank padding versus ignoring trailing blanks) yields the same results. However, there are differences for inequality comparisons. For example, suppose you have the two-byte string value 'a*' where the '*' represents a character in the database's collation that is less than the value of a blank. In previous versions of SQL Anywhere, the comparison predicate `'a*' < 'a'` returns TRUE. In version 10, the predicate yields FALSE, since the shorter string is not padded with blanks before being compared.

For more information about blank padding, see "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*] and "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

- **Case of return values for properties** Server properties (returned by the PROPERTY function) that returned YES or NO in previous releases now return Yes or No. Database properties (returned by the DB_PROPERTY function) and connection properties (returned by the CONNECTION_PROPERTY function) that returned ON or OFF in previous releases now return On or Off. This change may affect case-sensitive databases or applications that use case-sensitive string comparisons.

- **Changes to server property return values** In previous releases, the ConnsDisabled and RememberLastStatement server properties returned the values ON and OFF. They now return the values Yes and No. See "Database server properties" [*SQL Anywhere Server - Database Administration*].

- **Default location of sasrv.ini file has been changed** The default location of *sasrv.ini* is now *%ALLUSERSPROFILE%\Application Data\SQL Anywhere 10* on Windows and *$HOME/.sqlanywhere10* on Unix. In previous releases, the Unix file was named *.sasrv.ini*. On all platforms, the file is now named *sasrv.ini*.

- **Connections to database servers with long names** On Windows and Unix, version 9.0.2 and earlier clients cannot connect to version 10.0.0 and later database servers with names longer than 40 bytes.

- **Character set conversion is always enabled on the database server** The -ct database server option for enabling and disabling character set conversion is no longer supported. Character set conversion is always enabled for the database server, but if the database server determines that it is not required, then it is not used. You can disable character set conversion from the client by specifying CharSet=none. See "CharSet (CS) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Character set conversion unsupported on Windows CE** Character set conversion is not supported on Windows CE. In previous releases, character set conversion was disabled on the database server for Windows CE, and you could use any character set for the database. Now you must create databases for Windows CE using either the operating system character set or UTF-8. See "Installation considerations: Using ICU on Windows Mobile" [*SQL Anywhere Server - Database Administration*].

- **Changes to system procedures and functions** Following is a list of changes to system procedures and functions:

- **Several system procedures have been made internal** The external system procedures xp_read_file, xp_write_file, xp_sprintf, xp_scanf, and xp_cmdshell are now internal system procedures.
- **sa_validate system procedure** The sa_validate system procedure now requires VALIDATE authority. See "sa_validate system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_reset_identity system procedure** The table-name parameter is now required. Additionally, if the owner-name parameter is not specified, the table-name parameter must uniquely identify a table in the database. See "sa_reset_identity system procedure" [*SQL Anywhere Server - SQL Reference*].
- **sa_locks system procedure** The output of the sa_locks system procedure has been changed to return additional information, including the connection ID, the user ID, the table name, the lock class, and lock duration. See "sa_locks system procedure" [*SQL Anywhere Server - SQL Reference*].
- **RAND function** Previously, each connection was seeded with the same value so that the RAND function would return identical sequences for each connection. Now, each connection is uniquely seeded so that each connection will see a different random sequence. See "RAND function [Numeric]" [*SQL Anywhere Server - SQL Reference*].
- **DB_CALLBACK_START and DB_CALLBACK_FINISH callback functions** The DB_CALLBACK_START and DB_CALLBACK_FINISH callback functions are now supported on all platforms (previously, they were only supported on Windows platforms). See "db_register_a_callback function" [*SQL Anywhere Server - Programming*].

- **Scattered reads no longer used for files specified using a UNC name** Scattered reads are no longer used for files on remote computers, or for files specified using a UNC name such as *\\mycomputer\myshare\mydb.db*. See "Use an appropriate page size" [*SQL Anywhere Server - SQL Usage*].
- **Column ordering in primary and foreign key constraints** When creating primary key constraint, you can specify any order to the columns, regardless of the order in which columns appear in the table. Also, you can now create foreign keys that have a column order different from the primary key to which they refer, provided you specify the mapping between the foreign key columns and primary key columns. See the PRIMARY KEY clause in "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*].
- **Duplicate column names no longer allowed in indexes** Previously, duplicate references to columns in an index were allowed, except for primary key, foreign key, and unique constraint specifications. Now, the behavior is consistent across all types of indexes; specifying duplicate column names returns an error. Additionally, if an older database contains an index with duplicate column references, the dbunload utility drops the duplicate columns from the index when generating *reload.sql*. See "CREATE TABLE statement" [*SQL Anywhere Server - SQL Reference*].
- **Encryption database property** Executing `SELECT DB_PROPERTY( 'Encryption' )` may now return a value other than None, even when the database is not encrypted. This occurs when table encryption is enabled for the database. If you have an application that executes this command as a method for checking whether the database is encrypted, use `SELECT`

`DB_PROPERTY( 'EncryptionScope' )`, instead. See "Database properties" [*SQL Anywhere Server - Database Administration*].

- **Change in syntax for starting HTTPS using FIPS** Previously, you would specify `-xs HTTPS_FIPS(...)`. Now, you must specify `-xs HTTPS(FIPS=yes;...)`. The former syntax is still supported, but is deprecated. See "-xs dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Maximum user ID length is 128 bytes** In previous releases, when a statement required a user ID, the database server truncated user IDs longer than 128 bytes before using them in database server. If the string_rtruncation option was set, a truncation error was returned. The database server now returns an error if you specify a user ID that is longer than 128 bytes, regardless of the setting of the string_rtruncation option. See "Identifiers" [*SQL Anywhere Server - SQL Reference*].

- **Maximum length for server names** The maximum length of database server names has been increased from 40 bytes to 250 bytes on TCP/IP and shared memory connections. See "-n dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Changes to acceptable characters in identifiers** Double quotes and backslashes are no longer permitted in identifiers. See "Identifiers" [*SQL Anywhere Server - SQL Reference*].

- **LicensesInUse server property renamed** The server property LicensesInUse has been renamed to UniqueClientAddresses. See "Database server properties" [*SQL Anywhere Server - Database Administration*].

- **SQL Anywhere OLE DB provider name has changed** The SQL Anywhere OLE DB provider (previously, ASAProv, ASAProv.90, ASAProv.80) is now called SAOLEDB. The version 10 provider can be referenced specifically by the name SAOLEDB.10.

- **SQL Anywhere sample database ODBC DSN has changed** The ODBC data source (previously, ASA 9.0 Sample) is now called SQL Anywhere 10 Demo.

- **Changes to connection strings** For ODBC and OLE DB connections, the precedence of where a connection parameter is found is now: connection string, SQLCONNECT environment variable, data source. Previously, in ODBC and OLE DB the data source had higher precedence than SQLCONNECT. See "Connection parameter syntax rules" [*SQL Anywhere Server - Database Administration*].

- **Empty value connection parameters now treated as unspecified** For all APIs, connection parameters that are specified with empty values are treated as though the parameter was not specified. In previous releases, an empty value was treated as unspecified or as an empty string, depending on the location it was specified in and the API being used. See "Connection parameter syntax rules" [*SQL Anywhere Server - Database Administration*].

- **Transaction log cannot be turned off while auditing is on** In previous versions of the software, you could stop using the transaction log for a database that had auditing turned on. Now, you must use a transaction log if auditing is turned on for a database. You must turn auditing off if you want to stop using the transaction log.

- **Databases with auditing turned on cannot be started in read-only mode** In previous versions of the software, you could start databases in read-only mode with auditing turned on. Now, databases with auditing on cannot be started in read-only mode.

- **Precision of signed BIGINT columns now 19 instead of 20** Previously, when an ODBC application described a *signed* BIGINT column using SQL_BIGINT, a precision value of 20 was returned, which was incorrect. Now, a value of 19 is returned. You need to change any applications that relied on the previous (incorrect) value.

- **Java VM enhancements** SQL Anywhere no longer offers the Java option as a separately licensed component. Java in the database now uses an external VM to run your Java code instead of using an internal VM. As a result, you can now use any Java VM you want and you are no longer restricted to particular JDK versions or Java targets. Newly-initialized databases are always Java enabled.

  This results in the following changes:

  - **Unsupported database options** The following options are no longer supported in SQL Anywhere:

    - describe_java_format
    - java_heap_size
    - java_namespace_size
    - java_page_buffer_size
    - java_input_output
    - return_java_as_string

  - **Unsupported properties** Support has been removed for the following properties:

    - Database properties:

      - JDKVersion
      - JavaHeapSize
      - JavaNSSize

    - Database server properties:

      - IsJavaAvailable
      - JavaGlobFix

    - Connection properties:

      - JavaHeapSize
      - java_input_output

  - **New JavaVM property** The JavaVM database property returns the path to the Java VM that the database server uses to execute Java in the database.

- **Unsupported compatibility view columns** The following columns are no longer available in the system compatibility views:
  - SYSINFO.classes_version
  - SYSJAVACLASS.replaced_by
  - SYSJAVACLASS.type_id
- **Java options deprecated for database utilities** The following database utility options have been deprecated:
  - Initialization utility (dbinit): -ja, -jdk
  - Unload utility (dbunload): -jr
  - Upgrade utility (dbupgrad): -ja, -jdk, -jr, -j
- **Java support deprecated for some Java-related clauses in the CREATE DATABASE and ALTER DATABASE statements** The CREATE DATABASE statement no longer supports the JAVA ON | OFF and JDK version clauses, while the ALTER DATABASE statement no longer supports the REMOVE JAVA clause.
- **New Java file** In addition to the changes mentioned, the following file has been added: *java \sajvm.jar*.

- **Ping utility (dbping)** Previously, the Ping utility (dbping) reported an error if the database server returned NULL for a property value. Now, dbping prints NULL when a property value is unknown and exits with a success return code. You can specify the -en option if you want dbping to exit with a failed return code when a property value is unknown. See "Ping utility (dbping)" [*SQL Anywhere Server - Database Administration*].

- **Environment variables renamed** The following environment variables have been renamed for this release:

Previous name	New name
ASTMP	SATMP
ASDIR	SADIR
ASLOGDIR	SALOGDIR
ASLANG	SALANG
ASCHARSET	SACHARSET

- **Changes to PHP module file names** The naming convention for the PHP module files has been changed. In previous versions, the files were named *phpX_sqlanywhereY.dll*, where *X* was the PHP major version number and *Y* was the major SQL Anywhere version number. The PHP module files are now named *php-a.b.c_sqlanywhereY.dll*, where *a.b.c* is the full version number of the PHP source the file is built against and *Y* is the major SQL Anywhere version number. For example, *php-5.0.2_sqlanywhere10.dll*.

- **Specifying values for the PrefetchBuffer connection parameter** The PrefetchBuffer connection parameter now interprets values less than 16384 as kilobytes for backward compatibility. Using kilobytes without the k suffix is deprecated. If the value of PrefetchBuffer is adjusted because it was out of the valid range or specified in kilobytes without the k suffix, the client log file shows the actual PrefetchBuffer value used. See "PrefetchBuffer (PBUF) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **System-defined domains cannot be dropped** You can no longer drop system-defined domains, such as MONEY or UNIQUEIDENTIFIERSTR, from a database. See "DROP DOMAIN statement" [*SQL Anywhere Server - SQL Reference*].

- **Changes to database utilities** Following is a list of changes to the database utilities, as described:

  - **The Service utility (dbsvc) can grant the Login as a Service privilege** The Service utility (dbsvc) prompts you to grant the Login as a Service privilege if the -a option is used and you try to run a service under an account that does not have the Login as a Service privilege enabled. If the -y option is used, dbsvc attempts to grant the Login as a Service privilege without prompting you. See "Service utility (dbsvc) for Windows" [*SQL Anywhere Server - Database Administration*].

  - **The Unload utility (dbunload) -an option can be used against a remote server** Before this change you could only run dbunload -an against a server on the same computer. Now you can run dbunload -an against a server that is running on a different computer. See "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

  - **The Server Enumeration utility (dblocate) host name or IP address formats** The host name or IP address can use any format, regardless of whether -n is specified. For example, if a server is running on myhost.mycompany.com, which has IP address of 1.2.3.4, to list only servers running on this computer from any computer with the mycompany.com domain, any of `dblocate myhost`, `dblocate myhost.mycompany.com`, or `dblocate 1.2.3.4` can be used. In previous versions, only `dblocate myhost.mycompany.com` or `dblocate -n 1.2.3.4` would have worked since the given hostname or IP address had to match the address string (excluding the port number) displayed by dblocate. See "Server Enumeration utility (dblocate)" [*SQL Anywhere Server - Database Administration*].

- **Default-related changes** The following changes have been made to defaults:

  - **Default TCP/IP listening address changed for personal database server** On Windows, the personal database server now listens for connections on 127.0.0.1, rather than 0.0.0.0. This change means that users running SQL Anywhere with Windows Firewall enabled do not need to add dbeng10 to the exception list before it can be used.

    As a result of this change, trying to connect with `LINKS=tcpip(HOST=`*`hostname`*`;DOBROADCAST=none)` will not work if *hostname* is the real host name or IP address of the computer. However, using a hostname of **localhost** or **127.0.0.1** will work.

  - **Default database page size changed to 4096** The default database page size for SQL Anywhere databases has been changed to 4096 bytes from 2048 bytes. This page size has been shown to improve performance in many environments. See "CREATE DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

If you do not specify the -gp option and start a database server with no databases loaded, the default page size on the database server is 4096.

- **Default maximum cache size changes** The default, maximum cache size on Windows (non-AWE) has been increased. The default maximum cache size is now limited to the lesser of:
  - `90% of (total_physical_memory - 4 MB)`, but no less than 2 MB
  - `(available address space - 512 MB)`
- **Unix cache size** The way the maximum cache size is calculated on Unix has changed. The default maximum cache size is now calculated as follows:
  - On 32-bit Unix platforms, it is the lesser of 90% of total physical memory or 1,834,880 KB.

    On 64-bit Unix platforms, it is the lesser of 90% of total physical memory and 8,589,672,320 KB.

  See "-ch dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].
- **Unix stored procedures** When upgrading existing Unix applications, if you are using the 64-bit database server, any existing external stored procedures must be changed to 64-bit.
- **Default size when converting NULL constants to NUMERIC or string data types** When converting a NULL constant to the NUMERIC data type, or to string data types such as CHAR, and VARCHAR, the length is now set to 0, instead of 32767.
- **Default URI for openxml system procedure has changed** When using the openxml system procedure, if a namespace declaration is not specified, then by default the prefix mp is bound to the Uniform Resource Identifier (URI). In previous releases of the software, this URI was urn:ianywhere-com:asa-xpath-metaprop. The default URI has been renamed to urn:ianywhere-com:sa-xpath-metaprop. See "openxml system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Changes to cache size percentage calculation for -c, -ch-, and -cl server options** When using P (percentage) with -c, -ch, or -cl, the system now calculates percentage against either the amount of physical system memory, or the amount of available address space, whichever is lower. This eliminates the risk of allocating more memory for the cache than is available for addressing. See "-c dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*], "-ch dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*], and "-cl dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Procedure_profiling server option renamed** The correct name of the server option that controls procedure profiling is now ProcedureProfiling. The previous form, Procedure_profiling, is still accepted, but will be unsupported in a future release. See "sa_server_option system procedure" [*SQL Anywhere Server - SQL Reference*].

- **TCP/IP port number does not need to be specified by clients connecting to a database server on HP-UX which is not using the default port** In previous versions of the software, if you started a database server on HP-UX, you had to specify a port number using the ServerPort

[PORT] protocol option if the default port (2638) was already in use or if you did not want to use the default port.

On HP-UX, the TCP/IP ServerPort protocol option is no longer required when multiple database servers are started on one machine. On Mac OS X, the TCP/IP ServerPort option must still be specified when starting a network server if a server is already running on the same computer. See "ServerPort (PORT) protocol option" [*SQL Anywhere Server - Database Administration*].

- **SOAP CONCRETE response renamed from ASADataSet to SimpleDataset** The CONCRETE response has been renamed from ASADataSet to SimpleDataset. See "CREATE SERVICE statement" [*SQL Anywhere Server - SQL Reference*].
- **Unload Database wizard behavior changes** You can no longer unload a database into an database version earlier than version 10. When you unload a version 9.0.2 or earlier database into a version 10 database, you cannot connect to database automatically once the rebuild completes.
- **Extract Database wizard behavior changes** You cannot extract version 9.0.2 and earlier databases. You must extract from a version 10 database.
- **-ui and -ux server options unsupported on Solaris** The -ui and -ux options are no longer supported on Solaris. They are still available on Linux.
- **Converting numeric data types** When converting a DOUBLE type to NUMERIC, SQL Anywhere now uses an algorithm that more precisely approximates the original DOUBLE value. With these changes, DOUBLE values with 15 or fewer significant digits are precisely converted to NUMERIC. Sometimes this may lead to different answers than previous versions of SQL Anywhere. See "Converting between numeric sets" [*SQL Anywhere Server - SQL Reference*].
- **sa_validate system procedure changes** The data, index, and full options for the sa_validate system procedure are no longer required and their use is deprecated. Unless you are requesting an express or checksum validation, the checks carried out using the former data, index, and full options are now performed by default. See "sa_validate system procedure" [*SQL Anywhere Server - SQL Reference*].
- **a_validate_type enumeration changes** The VALIDATE_DATA, VALIDATE_INDEX, and VALIDATE_FULL parameters for the a_validate_type enumeration are no longer required and their use is deprecated. The validations performed by these options are now performed by default when VALIDATE_NORMAL is specified. See "a_validate_type enumeration" [*SQL Anywhere Server - Programming*].
- **SQLPATH environment variable syntax change** The syntax for the SQLPATH environment variable has changed on Unix. In previous versions, the path elements were separated by semicolons (;) for all operating systems. In SQL Anywhere 10, the path elements are separated by colons (:) on Unix platforms, and by semicolons on other platforms.

Database option changes

- **Case sensitivity and database options** The SET OPTION statement and CONNECTION_PROPERTY function use case insensitive option names. However, in databases that

use a Turkish collation or are case sensitive, option names referenced in queries should be written using the case specified in "Alphabetical list of options" [*SQL Anywhere Server - Database Administration*].

In either of these situations, executing a query on SYSOPTION or a query like the following may not match any rows if the option name is used with the wrong case:

```
SELECT *
   FROM sa_conn_properties()
   WHERE propname = 'BLOCKING'
```

- **Using embedded SQL with ansi_blanks set to On** For embedded SQL with ansi_blanks set to On and a blank padded database, when you supply a value of data type DT_STRING, you must set the sqllen field to the length of the buffer containing the value (at least the length of the value plus space for the terminating null character).

  When a database is blank padded, the ansi_blanks option controls truncation warnings sent to the client if the expression being fetched is CHAR or NCHAR (not VARCHAR or NVARCHAR) and it is being fetched into a char or nchar (not VARCHAR or NVARCHAR) host variable. See "ansi_blanks option" [*SQL Anywhere Server - Database Administration*].

- **ansi_integer_overflow option default setting changed** When a new database is created, the default value for the ansi_integer_overflow database option is On. In previous versions of the software, the default value for this option was Off.

- **date_format option changes** The date_format option no longer supports the following values when specifying the format string:

  - **hh** two digit hours
  - **nn** two digit minutes
  - **ss[.ss..]** seconds and parts of a second
  - **aa** morning/afternoon indicator (A.M. or P.M., 12 hour clock)
  - **aaa[a...]** morning/afternoon indicator (A.M. or P.M., 12 hour clock)
  - **pp** afternoon indicator, if necessary (P.M., 12 hour clock)
  - **ppp[p...]** afternoon indicator, if necessary (P.M., 12 hour clock)

  Also, if the character data is multibyte, the length of each symbol now reflects the number of characters. For example, the 'mmm' symbol specifies a length of three characters for the month. In previous versions, the length of the symbol reflected the number of bytes. See "date_format option" [*SQL Anywhere Server - Database Administration*].

- **login_mode database option** The value Mixed is deprecated for the login_mode database option. Specify Standard,Integrated to allow both standard and integrated logins. See "login_mode option" [*SQL Anywhere Server - Database Administration*].

- **string_rtruncation option default setting changed** When a new database is created, the default value for the string_rtruncation database option is On. In previous versions of the software, the default value for this option was Off. See "string_rtruncation option" [*SQL Anywhere Server - Database Administration*].

  If you use the CAST function to truncate strings, the string_rtruncation database option must be set to Off; otherwise, there will be an error.

  It is recommended that you use the LEFT function to truncate strings. See "LEFT function [String]" [*SQL Anywhere Server - SQL Reference*].

- **temp_space_limit_check option default setting changed** The default setting for the temp_space_limit_check option has been changed to On. Now, by default, if a connection requests more than its quota of temporary file space, then the request fails and the error SQLSTATE_TEMP_SPACE_LIMIT is returned. See "temp_space_limit_check option" [*SQL Anywhere Server - Database Administration*].

- **timestamp_format option changes** The timestamp_format option no longer supports the use of French days and months. Also, if the character data is multibyte, the length of each symbol now reflects the number of characters. For example, the 'mmm' symbol specifies a length of three characters for the month. In previous versions, the length of the symbol reflected the number of bytes. See "timestamp_format option" [*SQL Anywhere Server - Database Administration*].

- **truncate_date_values option removed** The truncate_date_values option has been removed. In previous releases, this option allowed you to include a time in columns defined using the DATE data type. In this release, columns defined with DATE can only contain dates. If you want to store dates and times, use the TIMESTAMP data type. See "TIMESTAMP data type" [*SQL Anywhere Server - SQL Reference*].

Server option changes

- **-ec server option and -xs server option TLS syntax for strong encryption types has changed** The syntax for the strong encryption types in the -ec server option, the -xs server option, and the Encryption connection parameter has changed. There are now only 3 types available: none, simple, and tls. Instead of specifying the key exchange algorithm to use as the type, you now specify tls as the encryption type, and use a new protocol option, tls_type, to specify the algorithm. See "-ec dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*], "-xs dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*], and "Encryption (ENC) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **-os server option** In previous releases, the -os database server option renamed the log file to *current-filename.old*. This is now the behavior of the -on option. The -os database server option now specifies a maximum size for the output log, at which point the log is renamed. Previously, using -os would result in two log files, but now it results in an unlimited number of log files. See "-os dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*] and "-on dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

Catalog changes

The catalog has undergone major changes in version 10.0.0. The most significant change is that system tables have been renamed to include an I at the beginning of their name. If you attempt to access the system tables, you will receive a permission denied error. Information in the system tables is made available through system views. There is one system view per system table, and, for backward compatibility, the system view names coincide with the table names from previous versions of SQL Anywhere. For example, in 9.0.2, there was a system table called SYS.SYSARTICLE. In version 10.0.0 that system table is now called SYS.ISYSARTICLE, and a corresponding system view, SYS.SYSARTICLE.

The catalog now also contains consolidated views. These are views which provide commonly needed joins from two or more tables or views. Most of the consolidated views were present as system views in previous releases.

Some system tables and views have been deprecated or removed from the catalog. However, most compatibility views are provided.

The following table provides a complete mapping of the catalog from Adaptive Server Anywhere 9.0.2 to SQL Anywhere 10.0.0. The first column, **9.0.2 system table/view**, shows names of the 9.0.2 system tables, followed by a forward slash ('/'), and then the name of the 9.0.2 associated view(s). The middle column, **10.0.0 system table**, contains the 10.0.0 table name. The final column, **10.0.0 system view**, contains the associated 10.0.0 view name(s), as well as compatibility notes.

Note
A dash (-) in any of the columns indicates that there is no equivalent object. For example, a new table in the catalog for the 10.0.0 release results in a dash for the table in the 9.0.2 column.

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
DUMMY / -	DUMMY	-
RowGenerator / -	RowGenerator	-
SYSARTICLE / SYSARTICLES	ISYSARTICLE	"SYSARTICLE system view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility:"SYSARTICLES consolidated view" [*SQL Anywhere Server - SQL Reference*].
SYSARTICLECOL / SYSARTICLECOL	ISYSARTICLECOL	"SYSARTICLECOL system view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSARTICLECOLS consolidated view" [*SQL Anywhere Server - SQL Reference*].
SYSATTRIBUTE / -	ISYSATTRIBUTE	-

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSATTRIBUTE-NAME / -	ISYSATTRIBUTE-NAME	-
SYSCAPABILITY / SY-SCAPABILITIES	ISYSCAPABILITY	"SYSCAPABILITY system view" [*SQL Anywhere Server - SQL Reference*] "SYSCAPABILITIES consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSCAPABILITY-NAME / -	ISYSCAPABILITY-NAME	"SYSCAPABILITYNAME system view" [*SQL Anywhere Server - SQL Reference*]
- / SYSCATALOG		"SYSCATALOG consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSCHECK / -	ISYSCHECK	"SYSCHECK system view" [*SQL Anywhere Server - SQL Reference*]
- / SYSCOLAUTH	-	"SYSCOLAUTH consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSCOLLATION / -	-	"SYSCOLLATION compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSCOLLATIONMAP-PINGS / -	-	"SYSCOLLATIONMAPPINGS compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSCOLPERM / -	ISYSCOLPERM	"SYSCOLPERM system view" [*SQL Anywhere Server - SQL Reference*]
SYSCOLSTAT / SY-SCOLSTATS	ISYSCOLSTAT	"SYSCOLSTAT system view" [*SQL Anywhere Server - SQL Reference*] and "SYSCOLSTATS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSCOLUMN / SY-SCOLUMNS	ISYSTABCOL	"SYSTABCOL system view" [*SQL Anywhere Server - SQL Reference*] and "SYSCOLUMNS consolidated view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSCOLUMN compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSCONSTRAINT / -	ISYSCONSTRAINT	"SYSCONSTRAINT system view" [*SQL Anywhere Server - SQL Reference*]
- / -	ISYSDEPENDENCY	"SYSDEPENDENCY system view" [*SQL Anywhere Server - SQL Reference*]
SYSDOMAIN / -	ISYSDOMAIN	"SYSDOMAIN system view" [*SQL Anywhere Server - SQL Reference*]
SYSEVENT / -	ISYSEVENT	"SYSEVENT system view" [*SQL Anywhere Server - SQL Reference*]
SYSEVENTTYPE / -	ISYSEVENTTYPE	"SYSEVENTTYPE system view" [*SQL Anywhere Server - SQL Reference*]
SYSEXTENT / -	-	-
SYSEXTERNLOGINS / -	ISYSEXTERNLOGIN	"SYSEXTERNLOGIN system view" [*SQL Anywhere Server - SQL Reference*]
SYSFILE / -	ISYSFILE	"SYSFILE compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSFKCOL / -	ISYSIDXCOL	"SYSIDXCOL system view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSFKCOL compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSFOREIGNKEY / SYSFOREIGNKEYS	ISYSFKEY	"SYSFKEY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSFOREIGNKEYS consolidated view" [*SQL Anywhere Server - SQL Reference*]. For pre-10.0.0 compatibility: "SYSFOREIGNKEY compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
- / SYSGROUPS	ISYSGROUP	"SYSGROUP system view" [*SQL Anywhere Server - SQL Reference*] and "SYSGROUPS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSHISTORY / -	ISYSHISTORY	"SYSHISTORY system view" [*SQL Anywhere Server - SQL Reference*]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSINDEX / SYSINDEXES	ISYSIDX	"SYSIDX system view" [*SQL Anywhere Server - SQL Reference*] and "SYSINDEXES consolidated view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSINDEX compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSINFO / -	-	"SYSINFO compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSIXCOL / -	ISYSIDXCOL	"SYSIDXCOL system view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSIXCOL compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSJAR / -	ISYSJAR	"SYSJAR system view" [*SQL Anywhere Server - SQL Reference*]
SYSJARCOMPONENT / -	ISYSJARCOMPONENT	"SYSJARCOMPONENT system view" [*SQL Anywhere Server - SQL Reference*]
SYSJAVACLASS / -	ISYSJAVACLASS	"SYSJAVACLASS system view" [*SQL Anywhere Server - SQL Reference*]
SYSLOGIN / -	ISYSLOGINMAP	"SYSLOGINMAP system view" [*SQL Anywhere Server - SQL Reference*]
SYSOPTBLOCK / -	-	system use only
- / -	ISYSMVOPTION	"SYSMVOPTION system view" [*SQL Anywhere Server - SQL Reference*]
- / -	ISYSMVOPTIONNAME	"SYSMVOPTIONNAME system view" [*SQL Anywhere Server - SQL Reference*]
- / -	ISYSOBJECT	"SYSOBJECT system view" [*SQL Anywhere Server - SQL Reference*]
SYSOPTION / SYSOPTIONS	ISYSOPTION	"SYSOPTION system view" [*SQL Anywhere Server - SQL Reference*] and "SYSOPTIONS consolidated view" [*SQL Anywhere Server - SQL Reference*]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
SYSOPTJOINSTRATEGY / SYSOPTJOINSTRATEGIES	-	system use only
SYSOPTORDER / SYSOPTORDERS	-	system use only
SYSOPTQUANTIFIER / -	-	system use only
SYSOPTREQUEST / -	-	system use only
SYSOPTREWRITE / -	-	system use only
SYSOPTSTAT / -	ISYSOPTSTAT	"SYSOPTSTAT system view" [*SQL Anywhere Server - SQL Reference*]
-	ISYSPHYSIDX	"SYSPHYSIDX system view" [*SQL Anywhere Server - SQL Reference*]
- / SYSPROCAUTH	-	"SYSPROCAUTH consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSPROCEDURE / SYSPROCEDURES	ISYSPROCEDURE	"SYSPROCEDURE system view" [*SQL Anywhere Server - SQL Reference*] The SYSPROCEDURES view has been renamed to SYSPROCS. See "SYSPROCS consolidated view" [*SQL Anywhere Server - SQL Reference*].
SYSPROCPARM / SYSPROCPARMS	ISYSPROCPARM	"SYSPROCPARM system view" [*SQL Anywhere Server - SQL Reference*] and "SYSPROCPARMS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSPROCPERM / -	ISYSPROCPERM	"SYSPROCPERM system view" [*SQL Anywhere Server - SQL Reference*]
-	ISYSPROXYTAB	"SYSPROXYTAB system view" [*SQL Anywhere Server - SQL Reference*]
SYSPUBLICATION / SYSPUBLICATIONS	ISYSPUBLICATION	"SYSPUBLICATION system view" [*SQL Anywhere Server - SQL Reference*] and "SYSPUBLICATIONS consolidated view" [*SQL Anywhere Server - SQL Reference*]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
- / -	ISYSREMARK	"SYSREMARK system view" [*SQL Anywhere Server - SQL Reference*]
SYSREMOTEOPTION / SYSREMOTEOP-TIONS, SYSREMO-TEOPTION2	ISYSREMOTEOP-TION	"SYSREMOTEOPTION system view" [*SQL Anywhere Server - SQL Reference*], "SYSRE-MOTEOPTION2 consolidated view" [*SQL Anywhere Server - SQL Reference*], and "SYSRE-MOTEOPTIONS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSREMOTEOPTION-TYPE / -	ISYSREMOTEOP-TIONTYPE	"SYSREMOTEOPTIONTYPE system view" [*SQL Anywhere Server - SQL Reference*]
SYSREMOTETYPE / SYSREMOTETYPES	ISYSREMOTETYPE	"SYSREMOTETYPE system view" [*SQL Anywhere Server - SQL Reference*] and "SYSRE-MOTETYPES consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSREMOTEUSER / SYSREMOTEUSERS	ISYSREMOTEUSER	"SYSREMOTEUSER system view" [*SQL Anywhere Server - SQL Reference*] and "SYSRE-MOTEUSERS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSSCHEDULE / -	ISYSSCHEDULE	"SYSSCHEDULE system view" [*SQL Anywhere Server - SQL Reference*]
SYSSERVERS / -	ISYSSERVER	"SYSSERVER system view" [*SQL Anywhere Server - SQL Reference*]
- / -	ISYSSOURCE	"SYSSOURCE system view" [*SQL Anywhere Server - SQL Reference*]
SYSSQLSERVER-TYPE / -	ISYSSQLSERVER-TYPE	"SYSSQLSERVERTYPE system view" [*SQL Anywhere Server - SQL Reference*]
SYSSUBSCRIPTION / SYSSUBSCRIPTIONS	ISYSSUBSCRIPTION	"SYSSUBSCRIPTION system view" [*SQL Anywhere Server - SQL Reference*] and "SYS-SUBSCRIPTIONS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSSYNC / SYS-SYNCS, SYSSYNC2	ISYSSYNC	"SYSSYNC system view" [*SQL Anywhere Server - SQL Reference*], "SYSSYNC2 consolidated view" [*SQL Anywhere Server - SQL Reference*], and "SYSSYNC2 consolidated view" [*SQL Anywhere Server - SQL Reference*]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
-	ISYSSYNCSCRIPT	"SYSSYNCSCRIPT system view" [*SQL Anywhere Server - SQL Reference*] and "SYSSYNCSCRIPTS consolidated view" [*SQL Anywhere Server - SQL Reference*]
- / SYSSYNCSUBSCRIPTIONS	-	"SYSSYNCSUBSCRIPTIONS consolidated view" [*SQL Anywhere Server - SQL Reference*]
- / SYSSYNCUSERS	-	"SYSSYNCUSERS consolidated view" [*SQL Anywhere Server - SQL Reference*]
- / SYSTABAUTH	-	"SYSTABAUTH consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSTABLE / -	ISYSTAB	"SYSTAB system view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSTABLE compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
-	ISYSTABCOL	"SYSTABCOL system view" [*SQL Anywhere Server - SQL Reference*]
SYSTABLEPERM / -	ISYSTABLEPERM	"SYSTABLEPERM system view" [*SQL Anywhere Server - SQL Reference*]
SYSTRIGGER / SYSTRIGGERS	ISYSTRIGGER	"SYSTRIGGER system view" [*SQL Anywhere Server - SQL Reference*] and "SYSTRIGGERS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSTYPEMAP / -	ISYSTYPEMAP	"SYSTYPEMAP system view" [*SQL Anywhere Server - SQL Reference*]
-	ISYSUSER	"SYSUSER system view" [*SQL Anywhere Server - SQL Reference*]
- / SYSUSERAUTH	ISYSUSERAUTHORITY	"SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERAUTH compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]

9.0.2 system table/view	10.0.0 system table	10.0.0 system view
- / SYSUSERLIST		"SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERLIST compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSUSERMESSAGES / -	ISYSUSERMESSAGE	"SYSUSERMESSAGE system view" [*SQL Anywhere Server - SQL Reference*]
- / SYSUSEROPTIONS	-	"SYSUSEROPTIONS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSUSERPERM / SYSUSERPERMS	-	Data now located in the ISYSUSER and ISYSUSERAUTHORITY system tables. See: "SYSUSER system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] For pre-10.0.0 compatibility: "SYSUSERPERM compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERPERMS compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*]
SYSUSERTYPE / -	ISYSUSERTYPE	"SYSUSERTYPE system view" [*SQL Anywhere Server - SQL Reference*]
- / SYSVIEWS	ISYSVIEW	"SYSVIEW system view" [*SQL Anywhere Server - SQL Reference*] and "SYSVIEWS consolidated view" [*SQL Anywhere Server - SQL Reference*]
SYSWEBSERVICE / -	ISYSWEBSERVICE	"SYSWEBSERVICE system view" [*SQL Anywhere Server - SQL Reference*]

Summary of new views

System view name	Link to more information
SYSDEPENDENCY	Each row in the SYSDEPENDENCY system view describes a dependency between two database objects. See "SYSDEPENDENCY system view" [*SQL Anywhere Server - SQL Reference*]

System view name	Link to more information
SYSFKEY	Each row in the SYSFKEY system view describes a foreign key constraint in the system. See "SYSFKEY system view" [*SQL Anywhere Server - SQL Reference*].
SYSIDX	Each row in the SYSIDX system table defines a logical index in the database. See "SYSIDX system view" [*SQL Anywhere Server - SQL Reference*].
SYSIDXCOL	Each row in the SYSIDXCOL system view describes one column of an index described in the SYSIDX system view. See "SYSIDXCOL system view" [*SQL Anywhere Server - SQL Reference*].
SYSLOGINMAP	The SYSLOGINMAP system view contains all the user names that can be used to connect to the database using either an integrated login, or Kerberos login. See "SYSLOGINMAP system view" [*SQL Anywhere Server - SQL Reference*].
SYSMVOPTION	Each row in the SYSMVOPTION system view describes the setting of one option value for a materialized view. See "SYSMVOPTION system view" [*SQL Anywhere Server - SQL Reference*].
SYSMVOPTION-NAME	Each row in the SYSMVOPTIONNAME system view contains the name of an option defined in the SYSMVOPTION system view. See "SYSMVOPTIONNAME system view" [*SQL Anywhere Server - SQL Reference*].
SYSOBJECT	Each row in the SYSOBJECT system view describes an object. Examples of database objects include tables, views, columns, indexes, and procedures. See "SYSOBJECT system view" [*SQL Anywhere Server - SQL Reference*].
SYSPHYSIDX	Each row in the SYSPHYSIDX system view defines a physical index in the database. See "SYSPHYSIDX system view" [*SQL Anywhere Server - SQL Reference*].
SYSPROCS	The SYSPROCS system view replaces the former SYSPROCEDURES view. See "SYSPROCS consolidated view" [*SQL Anywhere Server - SQL Reference*].
SYSPROXYTAB	Each row of the SYSPROXYTAB system view describes the remote parameters of one proxy table. See "SYSPROXYTAB system view" [*SQL Anywhere Server - SQL Reference*].
SYSREMARK	Each row in the SYSREMARK system view describes a remark (or comment) for an object. See "SYSREMARK system view" [*SQL Anywhere Server - SQL Reference*].

System view name	Link to more information
SYSSOURCE	Each row in the SYSSOURCE system view contains the source for an object listed in the ISYSOBJECT system table. See "SYSSOURCE system view" [*SQL Anywhere Server - SQL Reference*].
SYSSYNCSCRIPT	Each row in the SYSSYNCSCRIPT system view identifies a stored procedure for MobiLink scripted upload. See "SYSSYNCSCRIPT system view" [*SQL Anywhere Server - SQL Reference*].
SYSTABCOL	The SYSTABCOL system view contains one row for each column of each table and view in the database. See "SYSTABCOL system view" [*SQL Anywhere Server - SQL Reference*].
SYSUSER	Each row in the SYSUSER system view describes a user in the database. See "SYSUSER system view" [*SQL Anywhere Server - SQL Reference*].
SYSUSERAUTHORITY	Each row of SYSUSERAUTHORITY system view describes an authority granted to one user ID. See "SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*].

Summary of deprecated tables or views

Following is a list of catalog objects that are deprecated. Usually, the object in a table in previous versions is now a compatibility view. Referencing these objects does not result in an error; however, for future compatibility, you are encouraged to change your applications to point to the suggested object(s) instead.

Deprecated table or view	Transition information
SYSCOLLATION system table	Collation mapping information is now stored as database properties. See "SYSCOLLATION compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSCOLLATIONMAPPINGS system table	Collation mapping information is now stored as database properties. See "SYSCOLLATIONMAPPINGS compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSCOLUMN system table	Use the SYSTABCOL system view instead. See "SYSTABCOL system view" [*SQL Anywhere Server - SQL Reference*] and "SYSCOLUMN compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSFKCOL system table	Use the SYSFKEY system view instead. See "SYSFKEY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSFKCOL compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].

Deprecated table or view	Transition information
SYSFOREIGNKEY system table	Use the SYSFKEY system view instead. See "SYSFKEY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSFOREIGNKEY compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSINDEX system table	Use the SYSIDX system view instead. See "SYSIDX system view" [*SQL Anywhere Server - SQL Reference*] and "SYSINDEX compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSIXCOL system table	Use the SYSIDXCOL system view instead. See "SYSIDXCOL system view" [*SQL Anywhere Server - SQL Reference*] and "SYSIXCOL compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSTABLE system table	Use the SYSTAB system view instead. See "SYSTAB system view" [*SQL Anywhere Server - SQL Reference*] and "SYSTABLE compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSUSERAUTH system view	Use the SYSUSERAUTHORITY system view instead. See "SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERAUTH compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSUSERPERM system table	Use the SYSUSERAUTHORITY system view instead. See "SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERPERM compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSUSERLIST system view	Use the SYSUSERAUTHORITY system view instead. See "SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERLIST compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].
SYSUSERPERMS system view	Use the SYSUSERAUTHORITY system view instead. See "SYSUSERAUTHORITY system view" [*SQL Anywhere Server - SQL Reference*] and "SYSUSERPERMS compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].

Summary of removed or renamed tables or views

Following is a list of catalog objects that are no longer present in the catalog. Referencing these objects results in an error.

Removed table or view	Transition information
SYSATTRIBUTE system table	Use the SYSTAB and SYSPHYSIDX system views instead. Information about percent free and clustered index is now maintained in the ISYSTAB system table. Information about key values, key distance, leaf pages, and depth is now stored in the ISYSPHYSIDX system table. See "SYSTAB system view" [*SQL Anywhere Server - SQL Reference*] and "SYSPHYSIDX system view" [*SQL Anywhere Server - SQL Reference*].
SYSATTRIBUTENAME system table	Use the SYSIDX and SYSPHYSIDX system views instead. See "SYSIDX system view" [*SQL Anywhere Server - SQL Reference*] and "SYSPHYSIDX system view" [*SQL Anywhere Server - SQL Reference*].
SYSEXTENT system table	The SYSEXTENT table is no longer available in the catalog in SQL Anywhere version 10.0.0 and later. This table was previously unused.
SYSEXTERNLOGINS	Renamed to SYSEXTERNLOGIN. See "SYSEXTERNLOGIN system view" [*SQL Anywhere Server - SQL Reference*].
SYSLOGIN system table	The SYSLOGIN table has been replaced by the SYSLOGINMAP system view, with some changes. See "SYSLOGINMAP system view" [*SQL Anywhere Server - SQL Reference*].
SYSOPTBLOCK	This table was for internal use only.
SYSOPTJOINSTRATEGY	This table was for internal use only.
SYSOPTJOINSTRATEGIES	This view was for internal use only.
SYSOPTORDER	This table was for internal use only.
SYSOPTORDERS	This view was for internal use only.
SYSOPTQUANTIFIER	This table was for internal use only.
SYSOPTREQUEST	This table was for internal use only.
SYSOPTREWRITE	This table was for internal use only.
SYSPROCEDURES view	Use the SYSPROCS consolidated view instead. See "SYSPROCS consolidated view" [*SQL Anywhere Server - SQL Reference*].
SYSSERVERS	Renamed to SYSSERVER. See "SYSSERVER system view" [*SQL Anywhere Server - SQL Reference*].
SYSUSERMESSAGES	Renamed to SYSUSERMESSAGE. See "SYSUSERMESSAGE system view" [*SQL Anywhere Server - SQL Reference*].

Change to columns in system tables and system views

There have been numerous changes to columns in system tables and views. With the exception of the information below, all of the changes consist of adding new columns, or removing unused columns, neither of which impact your applications.

- **SYSCOLUMN and SYSCOLUMNS views** The width column for both of these views has changed from a SMALLINT to an UNSIGNED INT. See "SYSCOLUMN compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*] and "SYSCOLUMNS consolidated view" [*SQL Anywhere Server - SQL Reference*].

- **SYSCONSTRAINT view** The previous SYSCONSTRAINT system table has been replaced by a new system table, ISYSCONSTRAINT, with a corresponding SYSCONSTRAINT system view. References to SYSCONSTRAINT now use the new system view, which is significantly different in this release. To see the contents of the SYSCONSTRAINT system view, see "SYSCONSTRAINT system view" [*SQL Anywhere Server - SQL Reference*].

- **SYSREMOTEOPTION view** You can no longer select from SYSREMOTEOPTION. Use SYSREMOTEOPTIONS or SYSREMOTEOPTION2 instead. See "SYSREMOTEOPTIONS consolidated view" [*SQL Anywhere Server - SQL Reference*] or "SYSREMOTEOPTION2 consolidated view" [*SQL Anywhere Server - SQL Reference*].

- **SYSJAR, SYSJARCOMPONENT, and SYSJAVACLASS views** The create_time column has been removed. However the creation time information is available in SYSOBJECT.create_time. See "SYSOBJECT system view" [*SQL Anywhere Server - SQL Reference*].

- **SYSFILE system view** The store_type column is now an INTEGER. See "SYSFILE compatibility view (deprecated)" [*SQL Anywhere Server - SQL Reference*].

- **SYSPROCPARM and SYSLOGINMAP views** The remarks column has been removed from these views. Also, the width column in SYSPROCPARM has changed from a SMALLINT to an UNSIGNED INT. See "SYSPROCPARM system view" [*SQL Anywhere Server - SQL Reference*] and "SYSLOGINMAP system view" [*SQL Anywhere Server - SQL Reference*].

- **SYSPROCPARMS view** SYSPROCPARM.width has changed from a SMALLINT to an UNSIGNED INT. See "SYSPROCPARMS consolidated view" [*SQL Anywhere Server - SQL Reference*].

- **SYSREMOTEUSER view** The log_send, log_sent, confirm_sent, log_received, and confirm_received columns are now UNSIGNED BIGINT. See "SYSREMOTEUSER system view" [*SQL Anywhere Server - SQL Reference*].

- **SYSSUBSCRIPTION view** The created and started columns are now UNSIGNED BIGINT. See "SYSSUBSCRIPTION system view" [*SQL Anywhere Server - SQL Reference*].

- **SYSSYNC view** The progress, created, and log_sent columns are now UNSIGNED BIGINT. See "SYSSYNC system view" [*SQL Anywhere Server - SQL Reference*].

SQL statements

- **REVOKE CONNECT statement** When the REVOKE CONNECT statement is executed to drop a user, all objects owned by the specified user are dropped along with the user. The REVOKE CONNECT statement now returns an error if the database contains an active view, owned by another user, that is dependent upon an object owned by the user being dropped. See "REVOKE statement" [*SQL Anywhere Server - SQL Reference*].

- **Restrictions on key joins for derived tables** Key joins are not allowed for derived tables containing TOP N, START AT, FIRST, ORDER BY, window functions, FOR XML, or recursive tables. See "Key joins of views and derived tables" [*SQL Anywhere Server - SQL Usage*].

- **ALTER SERVER and CREATE SERVER statements** The ASAJDBC and ASAODBC server classes have been renamed to SAJDBC and SAODBC, respectively. See "ALTER SERVER statement" [*SQL Anywhere Server - SQL Reference*] and "CREATE SERVER statement" [*SQL Anywhere Server - SQL Reference*].

- **ALTER statements** All ALTER statements now use ALTER as a subclause, instead of MODIFY. If your applications use a MODIFY subclause, you should change them to use the ALTER subclause instead. The MODIFY syntax is still supported but deprecated. This impacts the following statements:

  - "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER EVENT statement" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER PUBLICATION statement [MobiLink] [SQL Remote]" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER SYNCHRONIZATION SUBSCRIPTION statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER SYNCHRONIZATION USER statement [MobiLink]" [*SQL Anywhere Server - SQL Reference*]
  - "ALTER TABLE statement" [*SQL Anywhere Server - SQL Reference*]

- **BACKUP statement** In previous releases, you could specify the DBFILE ONLY clause with either the TRANSACTION LOG RENAME or TRANSACTION LOG TRUNCATE clause. Specifying DBFILE ONLY with either of these TRANSACTION LOG clauses now results in an error because these are two mutually exclusive types of backup. See "BACKUP statement" [*SQL Anywhere Server - SQL Reference*].

- **COMMENT statement** The syntax COMMENT ON LOGIN is no longer supported. Use the syntax COMMENT ON INTEGRATED LOGIN instead. See "COMMENT statement" [*SQL Anywhere Server - SQL Reference*].

- **INSERT statement** In SQL Anywhere 10, when using the ON EXISTING SKIP and ON EXISTING ERROR clauses, if the table contains default columns, the server computes the default values even for rows that already exist. As a consequence, default values such as AUTOINCREMENT cause side effects even for skipped rows. In this case of AUTOINCREMENT, this results in skipped values in the AUTOINCREMENT sequence. In previous versions, these computations were not performed on default columns for skipped rows. See "INSERT statement" [*SQL Anywhere Server - SQL Reference*].

- **VALIDATE statements** All validation activities, such as executing VALIDATE statements, or running the Validation utility (dbvalid), now require VALIDATE authority; REMOTE DBA permission is no longer accepted for performing validation activities.

  The VALIDATE TABLE statement (and VALIDATE MATERIALIZED VIEW) checks for orphaned BLOBs.

  The syntax for VALIDATE INDEX has changed to be consistent with the ALTER INDEX statement syntax. The old syntax is still supported, although deprecated. If your applications currently use the VALIDATE INDEX statement, you should change to the new syntax.

  For more information on these changes, see "VALIDATE statement" [*SQL Anywhere Server - SQL Reference*].

Deprecated and discontinued features

- **-f, -fi, -fd, -fn options for the Validation utility (dbvalid) deprecated** The syntax for the dbvalid utility has been simplified. Previously, when no option was specified, the dbvalid utility performed express validation when validating tables. Now, the dbvalid utility performs full validation by default, as though the -f, -fi- and -fd options were specified. As a result, use of these options is deprecated and you must specify the -fx option to perform an express validation on tables.

  Also, support for the -fn option, which performed validation using the algorithm from version 9.0.0 and earlier releases, is no longer supported.

  For more information on the Validation utility, see "Validation utility (dbvalid)" [*SQL Anywhere Server - Database Administration*].

- **VALIDATE TABLE statement options deprecated** The syntax for the VALIDATE TABLE statement has been simplified. Previously, when no option was specified, the VALIDATE TABLE statement performed a normal validation. Now, the VALIDATE TABLE statement performs full validation by default, as though the WITH FULL CHECK options was specified. As a result, the WITH FULL CHECK, WITH INDEX, and WITH DATA options are deprecated. See "VALIDATE statement" [*SQL Anywhere Server - SQL Reference*].

- **Transact-SQL outer joins deprecated** Transact-SQL outer joins have been deprecated in this release, and will not be supported in future versions of SQL Anywhere. The new tsql_outer_joins database option enables or disables the ability to use the Transact-SQL outer joins operators *= and =* in DML statements and views for the current connection. This option is set to Off by default. See "tsql_outer_joins option" [*SQL Anywhere Server - Database Administration*].

- **WITH HASH SIZE clause no longer supported** With the elimination of older B-tree index technology, the WITH HASH SIZE clause of the CREATE INDEX statement is no longer supported.

- **Unsupported properties** The NumProcessorsAvail and NumProcessorsMax server properties are no longer supported. You can use the NumLogicalProcessors, NumLogicalProcessorsUsed,

NumPhysicalProcessors, and NumPhysicalProcessorsUsed server properties instead. See "Database server properties" [*SQL Anywhere Server - Database Administration*].

- **STRIP ON clause of LOAD TABLE is deprecated** While the stripping of leading and trailing blanks has been enhanced in SQL Anywhere 10.0.0 to allow you to fine tune the stripping behavior, STRIP ON is deprecated. To continue stripping trailing blanks *only* (default behavior in previous releases when STRIP ON was specified), use STRIP RTRIM instead. See "LOAD TABLE statement" [*SQL Anywhere Server - SQL Reference*].

- **UTF8 collation is deprecated** The UTF8 collation is deprecated. Use the UTF8BIN collation instead. See "Supported and alternate collations" [*SQL Anywhere Server - Database Administration*].

- **jConnect 4.5 no longer supported** Applications that previously connected using jConnect 4.5 will still work, but the using jConnect 5.5 or 6.0.5 is recommended. See "Using the jConnect JDBC driver" [*SQL Anywhere Server - Programming*].

- **SQLLOCALE environment variable no longer supported** The SQLLOCALE environment variable is no longer supported. It has been replaced by the SALANG and SACHARSET environment variables. See "SALANG environment variable" [*SQL Anywhere Server - Database Administration*] and "SACHARSET environment variable" [*SQL Anywhere Server - Database Administration*].

- **Named Pipes no longer supported** The Named Pipes protocol is no longer supported. Applications that previously used Named Pipes must be changed to use shared memory instead. See "Selecting communications protocols" [*SQL Anywhere Server - Database Administration*].

- **Data Source utility (dbdsn) -o option deprecated** The -o option for the Data Source utility has been deprecated. If you want to write output messages to a file, you can specify the LogFile connection parameter in the connection string. See "LogFile (LOG) connection parameter" [*SQL Anywhere Server - Database Administration*].

- **Creation of custom collations not supported** Creation of custom collations is no longer supported. The Create Custom Collation wizard, the Collation utility (dbcollat), the DBCollate function, and the a_db_collation structure, are no longer supported. See "Choosing collations" [*SQL Anywhere Server - Database Administration*].

  If you are rebuilding a database with a custom collation, the collation is preserved if you rebuild in a single step. If you choose to unload the database and then load the schema and data into a database that you create, then you must use one of the supplied collations. See "Rebuilding version 9 and earlier databases for version 12" on page 309.

- **Server Licensing utility -p option not supported** In previous releases, the Server Licensing utility supported the -p option, which was used to specify the operating system the database server was licensed for. This option is no longer supported.

- **Database server -d option not supported** The -d database server option, used on NetWare to force the use of POSIX I/O rather than DFS (Direct File System) I/O is no longer supported.

- **Database server -y option not supported** The -y database server option, used on Windows 95/98/Me to run the database server as a Windows service is no longer supported because these operating systems are no longer supported. To run the database server as a service on any of the supported

platforms, use the dbsvc utility. See "Service utility (dbsvc) for Windows" [*SQL Anywhere Server - Database Administration*].

- **-sc option not supported** SQL Anywhere 7.0 was awarded a TCSEC (Trusted Computer System Evaluation Criteria) C2 security rating from the U.S. Government. The -sc server option allowed you to run the current version of SQL Anywhere in a manner equivalent to the C2-certified environment. Support for the -sc option, as well as the C2 server property, has been version 10.0.0 removed.

- **max_work_table_hash_size database option not supported** The max_work_table_hash_size option is no longer supported. The query optimizer allocates hash sizes for the internal temporary tables based on the data distribution within the table.

- **max_hash_size database option not supported** The max_hash_size option is no longer supported.

- **Compressed databases and write files not supported** As a result, the following features are no longer available:

  - **File extensions** The following file extensions are no longer supported:

    - the *.wrt* extension used to identify write files

    - the *.cdb* extension used to identify compressed database files

  - **Database server behavior on NetWare** The database server no longer looks for database files with the *.wrt* extension when a database file is specified without an extension. See "The SQL Anywhere database server" [*SQL Anywhere Server - Database Administration*].

  - **Deploying databases on read-only media** You can no longer supply a write file to record changes to a database supplied on read-only media, such as a CD-ROM. However, you can still deploy databases on read-only media if they are run in read-only mode. See "Deploying databases on read-only media" [*SQL Anywhere Server - Programming*] and "-r dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

  - **Database utilities** The following utilities and wizards are no longer supported:

    - Compress Database wizard
    - Create Write File wizard
    - Uncompress Database wizard
    - Uncompression utility (dbexpand)
    - Compression utility (dbshrink)
    - Write File utility (dbwrite)

  - **SQL statements** The following SQL statements are no longer supported:

    - ALTER WRITEFILE
    - CREATE WRITEFILE
    - CREATE COMPRESSED DATABASE
    - CREATE EXPANDED DATABASE

- **DBTools structures** The following structures or members of structures are no longer supported:
  - **a_backup_db structure** This structure holds the information needed to perform backup tasks using the DBTools library.

    The backup_writefile member now appears as _unused.
  - **a_compress_db structure** This structure has been removed.
  - **a_compress_stats structure** This structure holds the information needed to perform database compression tasks using the DBTools library.
  - **a_db_info structure** This structure holds the information needed to return dbinfo information using the DBTools library.

    The wrtbufsize member now appears as _unused1, the wrtnamebuffer member now appears as _unused2, and the compressed member now appears as _unused3.
  - **an_expand_db structure** This structure holds information needed for database expansion using the DBTools library.
  - **a_stats_line structure** This structure holds information needed for database compression and expansion using the DBTools library.
  - **a_writefile structure** This structure holds information needed for database write file management using the DBTools library.
- **DBTools functions** The following functions are no longer supported:
  - DBChangeWriteFile
  - DBCompress
  - DBCreateWriteFile
  - DBExpand
  - DBStatusWriteFile
- **Database properties** The following database properties are no longer supported:
  - Compression
  - FileSize *writefile*
  - FreePages *writefile*
- **DB_BACKUP_WRITEFILE** This embedded SQL function is no longer supported.

- **Support for unused Adaptive Server Enterprise-compatibility views and procedures removed** Support for the following unused Adaptive Server Enterprise views in the SQL Anywhere database has been removed:

View name	View name
SYSALTERNATES	SYSLOGINROLES
SYSAUDITOPTIONS	SYSLOGS
SYSAUDITS	SYSMESSAGES
SYSCHARSETS	SYSPROCEDURES
SYSCONFIGURES	SYSPROCESSES
SYSCONSTRAINTS	SYSPROTECTS
SYSCURCONFIGS	SYSREFERENCES
SYSDATABASES	SYSREMOTELOGINS
SYSDEPENDS	SYSROLES
SYSDEVICES	SYSSEGMENTS
SYSENGINES	SYSSERVERS
SYSKEYS	SYSSRVROLES
SYSLANGUAGES	SYSTHRESHOLDS
SYSLOCKS	SYSUSAGES

Support for the following unused Adaptive Server Enterprise procedures in the SQL Anywhere database has been removed:

Procedure name	Procedure name
sp_addalias	sp_helpindex
sp_addauditrecord	sp_helpjoins
sp_addlanguage	sp_helpkey
sp_addremotelogin	sp_helplanguage
sp_addsegment	sp_helplog
sp_addserver	sp_helpremotelogin
sp_addthreshold	sp_helpprotect

Procedure name	Procedure name
sp_adddumpdevice	sp_helpsegment
sp_auditdatabase	sp_helpserver
sp_auditlogin	sp_helpsort
sp_auditobject	sp_helpthreshold
sp_auditoption	sp_helpuser
sp_auditsproc	sp_indsuspect
sp_bindefault	sp_lock
sp_bindmsg	sp_locklogin
sp_bindrule	sp_logdevice
sp_changedbowner	sp_modifylogin
sp_checknames	sp_modifythreshold
sp_checkreswords	sp_monitor
sp_clearstats	sp_placeobject
sp_commonkey	sp_primarykey
sp_configure	sp_procxmode
sp_cursorinfo	sp_recompile
sp_dboption	sp_remap
sp_dbremap	sp_remoteoption
sp_depends	sp_rename
sp_diskdefault	sp_renamedb
sp_displaylogin	sp_reportstats
sp_dropalias	sp_role
sp_dropdevice	sp_serveroption
sp_dropkey	sp_setlangalias

Procedure name	Procedure name
sp_droplanguage	sp_spaceused
sp_dropremotelogin	sp_syntax
sp_dropsegment	sp_unbindefault
sp_dropserver	sp_unbindmsg
sp_dropthreshold	sp_unbindrule
sp_estspace	sp_volchanged
sp_extendsegment	sp_who
sp_foreignkey	sp_column_privileges
sp_help	sp_databases
sp_helpconstraint	sp_datatype_info
sp_helpdb	sp_server_info
sp_helpdevice	sp_table_privileges
sp_helpgroup	

- **index_type and index_owner columns removed from SYSINDEX system view** The index_type and index_owner columns have been removed from the SYSINDEX view. These columns previously contained the default values USER and SA, respectively. Index information is now stored in the ISYSIDX and ISYSIDXCOL system views. See "SYSIDX system view" [*SQL Anywhere Server - SQL Reference*] and "SYSIDXCOL system view" [*SQL Anywhere Server - SQL Reference*].

- **DLL protocol option removed on server** The DLL protocol option now applies only to clients running on Windows 32-bit platforms. The DLL protocol option has been removed from the database server as it uses only Winsock 2.2. Similarly, the DLL protocol has been removed from Windows CE clients as they use only Winsock 1.1.

  Winsock 2.2 is required for database servers on all Windows platforms.

- **ASANY and ASANYSH environment variables renamed** The ASANY and ASANYSH environment variables have been renamed SQLANY10 and SQLANYSH10, respectively. See "SQLANY12 environment variable" [*SQL Anywhere Server - Database Administration*] and "SQLANYSAMP12 environment variable" [*SQL Anywhere Server - Database Administration*].

- **PreserveSource property deprecated** The PreserveSource database property has been deprecated in this release and always returns the value On when its setting is queried.

- **Unsupported database properties** The following database properties have been removed in this release:

  - BlobArenas
  - ClusteredIndexes
  - CompressedBTrees
  - FileVersion
  - FreePageBitMaps
  - Histograms
  - HistogramHashFix
  - IndexStatistics
  - LargeProcedureIDs
  - NamedConstraints
  - SeparateCheckpointLog
  - SeparateForeignKeys
  - StringHistogramsFix
  - TableBitMaps
  - TablesQualTriggers
  - TransactionsSpanLogs
  - UniqueIdentifier
  - VariableHashSize

- **Unsupported system procedures** The sa_conn_properties_by_name and sa_conn_properties_by_conn system procedures are no longer supported. You can use the new sa_conn_options system procedure to obtain this information. See "sa_conn_options system procedure" [*SQL Anywhere Server - SQL Reference*].

- **Algorithms removed from query optimization plans** The Lock, Nested Block Join, Sorted Block, and JNBO algorithms have been removed from query optimization plans, and lock nodes no longer appear in the plan. You can view locking information in the scan nodes in the plan.

- **util_db.ini file deprecated** Using the *util_db.ini* file to specify the password for the DBA user when connecting to the utility database has been deprecated. You can use the -su server option instead. See "Using util_db.ini with network database servers (deprecated)" [*SQL Anywhere Server - Database Administration*] and "-su dbeng12/dbsrv12 server option" [*SQL Anywhere Server - Database Administration*].

- **Deprecated Windows CE platforms** Support for Windows CE MIPS processors have been removed. For a list of supported platforms, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

MobiLink

The following sections describe the new features, behavior changes, and deprecated features in MobiLink for version 10.0.0.

New features

Following is a list of additions to MobiLink introduced in version 10.0.0.

Enhancements to the MobiLink plug-in for Sybase Central

It is now much easier to set up MobiLink applications by using a wizard to create a **synchronization model file**. This file contains information you enter about remote and consolidated tables and how to synchronize them. When the model is ready, you can use another wizard to deploy it, which will generate scripts and tables required for the application.

- **Create Synchronization Model wizard** With the new Create Synchronization Model wizard, you can quickly create and deploy MobiLink applications. This wizard can create a remote database or use existing remotes. It automates the creation of synchronization scripts, and can automatically handle download deletes, conflict resolution, and other challenging synchronization issues.

  See "Creating synchronization models" [*MobiLink - Getting Started*].

- **Model mode** After using the Create Synchronization Model wizard, you can use Model mode to customize your synchronization project before it is deployed. When you are in Model mode, you are working offline. Model mode stores your synchronization model as an XML file.

  See "Synchronization model tasks" [*MobiLink - Getting Started*].

- **Deploy wizard** When your model is customized, you can deploy it using the new Deploy wizard. The Deploy wizard adds the scripts, users, script versions, and so on to the MobiLink system tables on the consolidated database. Any changes you make to the consolidated database cannot be reengineered back to Model mode, although you can deploy the same model multiple times.

  See "Deploying synchronization models" [*MobiLink - Getting Started*].

- **Admin mode** The MobiLink plug-in as it existed before version 10.0.0 is now called Admin mode. Numerous enhancements have been made to Admin mode to make it easier to use. When you are in Admin mode, you are connected to your consolidated database and making changes live. You can use Admin mode to modify all your MobiLink consolidated databases.

Synchronize to any data source

A new feature called **direct row handling** allows you to synchronize to virtually any data source. For example, you can synchronize to application servers, web servers, web services, text files, Excel spreadsheets, J2ME devices, or an RDBMS that cannot be used as a consolidated database using SQL-based row handling. You must still have a consolidated database to hold MobiLink system tables and data that you want MobiLink to manage. The new data source or sources can be fully integrated into your synchronization process.

The MobiLink server API has been extended to support direct row handling, and two new events have been added. See:

- "Direct row handling" [*MobiLink - Server Administration*]
- "handle_DownloadData connection event" [*MobiLink - Server Administration*]
- "handle_UploadData connection event" [*MobiLink - Server Administration*]

In addition, a new feature called **mobile web services** provides support for mobile-optimized asynchronous web services that you can integrate with remote applications.

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

Easier deployment

There is now a Deployment wizard that you can use to deploy the MobiLink server, SQL Anywhere clients, the MobiLink Monitor, and encryption components. The InstallShield merge modules and templates that were provided with previous versions are no longer provided. See:

- "Using the Deployment Wizard" [*SQL Anywhere Server - Programming*]
- "Deploying MobiLink applications" [*MobiLink - Server Administration*]

Consolidated databases

New setup procedures

- **Setup script required by SQL Anywhere consolidated databases** You must now run a setup script before using a SQL Anywhere database as a MobiLink consolidated database, and the MobiLink system tables that are created with the setup script are now owned by the user who ran the setup script. This behavior is consistent with other consolidated database types. In previous versions of MobiLink, MobiLink system tables were owned by DBO in SQL Anywhere consolidated databases.

  See "SQL Anywhere consolidated database" [*MobiLink - Server Administration*].

- **Simplified setup procedures** You can now set up your consolidated database in Sybase Central or using setup scripts. Previously, you had to run a setup script. In addition, each type of consolidated database now has only one setup script. There are no more version-specific setup scripts (such as *syncase125.sql*).

  See "MobiLink consolidated databases" [*MobiLink - Server Administration*].

New system objects

- **New ways to clean up MobiLink system tables on your consolidated database** New system procedures have been added that help you do the following:

  - Purge information about obsolete remote databases from the MobiLink system tables.

    See "ml_delete_sync_state_before system procedure" [*MobiLink - Server Administration*].

  - Delete unused or unwanted synchronization state information.

    See "ml_delete_sync_state system procedure" [*MobiLink - Server Administration*].

- Reset synchronization state information.

  See "ml_reset_sync_state system procedure" [*MobiLink - Server Administration*].

- **New MobiLink server system tables and schema** Following are changes to the MobiLink system tables:

  - Several new MobiLink system tables have been added:

    - ml_database
    - ml_column
    - ml_qa_clients

  - The contents of ml_subscription are significantly different. The UltraLite synchronization sequence number, previously stored in ml_user.commmit_state, is now stored in ml_subscription.progress. The progress column also stores the SQL Anywhere remote synchronization progress.

  - The contents of ml_user are significantly different.

  - A checksum column has been added to the ml_script table.

  - The ml_user column of ml_listening has been changed to the name column.

  - A new system table has been added that is used internally by Sybase Central for server-initiated synchronization.

  - There have been changes to several QAnywhere system tables. See:

    - ml_qa_delivery
    - ml_qa_delivery_client
    - ml_qa_global_props
    - ml_qa_global_props_client
    - ml_qa_repository
    - ml_qa_repository_client
    - ml_qa_repository_props
    - ml_qa_repository_props_client
    - ml_qa_repository_staging

- **New system procedure ml_add_column** When you are using named row parameters, you may need to use this new system procedure to populate the ml_column MobiLink system table with information about columns on the remote database.

  See "ml_add_column system procedure" [*MobiLink - Server Administration*].

MobiLink server

New mlsrv10 features

- **Server name change to mlsrv10** The MobiLink server is now called mlsrv10. Previously, it was called dbmlsrv9.

- **New syntax for mlsrv10 -x** The mlsrv10 -x option, used for setting network protocol options for MobiLink clients, has changed.

  See "-x mlsrv12 option" [*MobiLink - Server Administration*].

- **New -xo option for older clients** To connect the MobiLink server to version 8 or 9 clients, you should use the mlsrv10 -xo option, which is identical to the dbmlsrv9 -x option. You can support version 8 and 9 clients, as well as version 10 clients, from one instance of mlsrv10, but to do so you need to use two different ports.

  The -xo option for HTTP and HTTPS includes a new option, session_key, which is useful if you cannot use JSESSIONID for tracking connections.

- **Improved handling of cache** The MobiLink server no longer maintains separate pools of memory for different tasks. All cache memory is shared by all synchronizations. You set the cache size using the new mlsrv10 -cm option. Other options for setting cache sizes (-bc, -d, -dd, and -u) have been removed.

  See "-cm mlsrv12 option" [*MobiLink - Server Administration*].

- **Ignore option now affects all streams** Now every host name or IP address that you specify to ignore is ignored on all -x streams. Previously (and still with -xo), the host was ignored only on the stream where it was specified.

  See "ignore" in "-x mlsrv12 option" [*MobiLink - Server Administration*].

- **Forcing upload scripts** The mlsrv10 -zus option allows you to force the MobiLink server to call upload scripts for a table, even when there is no data to upload for that table.

  See "-zus mlsrv12 option" [*MobiLink - Server Administration*].

- **New verbosity option** The new verbosity option **e** allows you to capture system event scripts. When -ve is specified, the MobiLink server shows all system event scripts that are used to maintain MobiLink system tables, as well as the SQL statements that define the upload stream.

  See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **File transfer directory** A new option has been added that allows you to use a directory for file transfers.

  See "-ftr mlsrv12 option" [*MobiLink - Server Administration*].

- **Set the maximum number of concurrent synchronizations** You can now improve performance by setting the maximum number of synchronizations that can be actively worked on.

See "-sm mlsrv12 option" [*MobiLink - Server Administration*].

- **Limit concurrent network connections** The new -nc option lets you specify a limit to the number of concurrent network connections.

  See "-nc mlsrv12 option" [*MobiLink - Server Administration*].

- **mlsrv10 now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the unambiguous ISO 8601 datetime format: **{I|W|E}***.yyyy-mm-dd hh:mm:ss message*.

New MobiLink scripting features

- **Named script parameters** There are now names for MobiLink event parameters. Previously, you had to specify script parameters as question marks. Now, question marks are optional. You can choose from a set of predefined named parameters, or create your own, or both. User-defined named parameters are useful when your RDBMS does not support variables. You can specify the named parameters in any order, and use any subset of available parameters, unlike question marks. Also, you can usually use the same named parameter multiple times in the same script.

  See "Script parameters" [*MobiLink - Server Administration*].

- **New conflict detection event** There is a new event that you can script to detect conflicts at the column level. This is an alternative to the upload_fetch event, which detects conflicts at the row level.

  See "upload_fetch_column_conflict table event" [*MobiLink - Server Administration*].

- **Global script version** You can now create a global script version. You define the scripts associated with the global script version once and then they are automatically used in all synchronizations unless you specify a script for the same event in the script version you are using to synchronize. When you are using multiple script versions, this means that you can avoid duplicating connection level scripts.

  See "ml_global script version" [*MobiLink - Server Administration*].

Performance enhancements

- **Improved MobiLink architecture** The MobiLink server has been re-architected to improve throughput, flexibility, and maintainability. The internal MobiLink client/server protocol has been enhanced for the same reasons.

  See "MobiLink performance" [*MobiLink - Server Administration*].

Other server enhancements

- **Snapshot isolation** For SQL Anywhere version 10 and Microsoft SQL Server 2005 and later consolidated databases, snapshot isolation is now the default for downloads, and is an option for uploads. MobiLink server options are added to help you control this behavior.

See:

- "MobiLink isolation levels" [*MobiLink - Server Administration*]
- "-dsd mlsrv12 option" [*MobiLink - Server Administration*]
- "-dt mlsrv12 option" [*MobiLink - Server Administration*]
- "-esu mlsrv12 option" [*MobiLink - Server Administration*]

- **Synchronization ID** Each synchronization is now identified by an integer that is between 1 and 4294967295. Each instance of the MobiLink server maintains its own synchronization IDs. When the MobiLink server is started, the ID is reset to 1. This ID is logged in the output file.

- **Improved MobiLink network layer** The network layer now includes compression, persistent connections (so you can synchronize multiple times on the same connection), IPv6 support, and improved error detection, liveness detection, and debugging.

MobiLink Monitor enhancements

- **Utility name change to mlmon** The MobiLink Monitor is now called mlmon. Previously, it was called dbmlmon.

  See "Starting the MobiLink Monitor" [*MobiLink - Server Administration*].

- **Multiple MobiLink Monitors** You can now connect multiple MobiLink Monitors to the same MobiLink server simultaneously. This allows multiple users to track synchronizations on the same server.

  See "Starting the MobiLink Monitor" [*MobiLink - Server Administration*].

- **Network options** The MobiLink Monitor now allows the same network options as MobiLink clients.

  See "Starting the MobiLink Monitor" [*MobiLink - Server Administration*].

- **New Utilization Graph** The Utilization Graph pane shows queue lengths within the MobiLink server.

  See "Utilization Graph pane" [*MobiLink - Server Administration*].

- **Viewing data in the Chart pane** In the Chart pane, you can still view data by user, or you can choose to view it in Compact view, which shows all active synchronizations in as few rows as possible. The Worker view has been removed because synchronization is no longer tied to a single worker thread.

  See "Chart pane" [*MobiLink - Server Administration*].

- **New Sample Properties window** The new Sample Properties shows data for a one-second interval or the average of all the one-second intervals in the selected period.

  See "Sample properties" [*MobiLink - Server Administration*].

- **Enhanced Session Properties window** Session properties now contains a detailed Statistics tab.

  See "Session properties" [*MobiLink - Server Administration*].

- **Ability to Monitor FIPS-enabled servers** The MobiLink Monitor can now monitor MobiLink servers that are running FIPS-approved encryption. Previously, it was not able to.

- **Changes to statistical properties** See "Changes to statistical properties" in "MobiLink server changes" on page 263.

MobiLink Redirector enhancements

- **Redirector supports groups of MobiLink servers** For some Redirectors, you can now create MobiLink server groups. Server groups can be used to support version 10 clients at the same time as version 8 or 9 clients through one Redirector, or for other purposes. For information about which Redirectors support server groups, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

  The Redirectors that support server groups have other enhancements to their configuration settings. In addition, they use a new sample configuration file called *redirector_server_group.config*.

- **HTTPS support** In previous versions of SQL Anywhere, when HTTPS is used for the connection between a remote database and web server, the web server decrypts the HTTPS and sends HTTP to MobiLink via the Redirector. Now, for some web servers, the Redirector re-encrypts the stream as HTTPS and sends it to the MobiLink server. There is new syntax for the ML directive in the Redirector configuration file. For information on which web servers have HTTPS support, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

Unix/Linux enhancements

- **MobiLink server messages window** Linux installations now have a messages window that shows log information for dbmlsync and mlsrv10.

  See "-ux dbmlsync option" [*MobiLink - Client Administration*] and "-ux mlsrv12 option" [*MobiLink - Server Administration*].

- **More consistent character conversions** There are improvements to the consistency of character conversions between Unix/Linux and Windows.

MobiLink clients

- **New remote ID** MobiLink now uses a new identifier called a remote ID to uniquely identify a remote database. Previous versions used the MobiLink user name. The remote ID is stored in the remote database. MobiLink generates a remote ID the first time a remote database synchronizes (or any time it encounters a NULL value for the remote ID). The remote ID is created automatically as a GUID, but you can set it to any string that has meaning to you. The remote ID lets the same MobiLink user synchronize multiple remote databases. In UltraLite remote databases, the remote ID is also useful for allowing multiple MobiLink users to synchronize the same remote database.

  Every script that accepts the MobiLink user name as a parameter now also accepts a remote_id parameter. The remote_id parameter is only available if you use named parameters.

  To help you change the remote ID, a new database option is added to both SQL Anywhere and UltraLite databases called ml_remote_id.

See:

- "Remote IDs" [*MobiLink - Client Administration*]
- "MobiLink user names and remote IDs" on page 270
- SQL Anywhere clients: "Setting remote IDs" [*MobiLink - Client Administration*]
- UltraLite clients: "UltraLite ml_remote_id option" [*UltraLite - Database Management and Reference*]

- **New file transfer functionality** New functionality helps you transfer files to remote devices using the same network path you use to synchronize data. SQL Anywhere clients can use the new mlfiletransfer utility, and UltraLite clients can use the new MLFileTransfer method. This functionality is especially useful when populating new remote databases or upgrading software. A new MobiLink event has been added to authenticate the file transfer, if desired. See:

  - SQL Anywhere clients: "MobiLink File Transfer utility (mlfiletransfer)" [*MobiLink - Client Administration*]
  - UltraLite clients: "Using MobiLink file transfers" [*UltraLite - Database Management and Reference*]
  - MobiLink server: "authenticate_file_transfer connection event" [*MobiLink - Server Administration*]

- **SendColumnNames has changed** The SendColumnNames dbmlsync extended option and Send Column Names UltraLite synchronization parameter were previously used to upload information about remote database columns so that the MobiLink server could generate sample synchronization scripts. The creation of sample synchronization scripts has been removed (and replaced with the Create Synchronization Model wizard). SendColumnNames is now used only by direct row handling. See:

  - "Direct row handling" [*MobiLink - Server Administration*]
  - "SendColumnNames (scn) extended option" [*MobiLink - Client Administration*]
  - "Send Column Names synchronization parameter" [*UltraLite - Database Management and Reference*]

- **Simplified liveness timeout settings** Liveness timeout is now controlled by the client. A new network protocol option called timeout is introduced that replaces liveness_timeout, contd_timeout, unknown_timeout, and network_connect_timeout.

  See "timeout" [*MobiLink - Client Administration*].

- **Buffer_size enhancements** Using the buffer_size network protocol option, you can now control write buffering for TCP/IP protocols as well as HTTP body size for the HTTP protocols. The default values have also changed.

  See "buffer_size" [*MobiLink - Client Administration*].

UltraLite clients

- **Palm support for network_leave_open** On Palm devices you can now choose whether network connectivity stays open after synchronization finishes. This functionality was available on other platforms in previous releases.

See "network_leave_open" [*MobiLink - Client Administration*].

- **UltraLite enhancements** For information on other UltraLite enhancements, see "Synchronization" on page 285.

SQL Anywhere clients

- **Scripted upload** In regular synchronization, dbmlsync uses the transaction log to create the upload, and so synchronizes all relevant data that has changed on the remote database since the last upload. You can now write stored procedures that define exactly what rows get uploaded, and so bypass the use of the transaction log. These stored procedures can perform DML and upload the result set, so the rows can be created dynamically, if required.

  See "Scripted upload" [*MobiLink - Client Administration*].

  Support for scripted uploads has required the following changes to SQL Anywhere system objects:

  - New column (sync_type) in the ISYSPUBLICATION system table.See "SYSPUBLICATION system view" [*SQL Anywhere Server - SQL Reference*].
  - New catalog objects in the ISYSSYNCSCRIPT system table for tracking synchronization scripts.

    See "SYSSYNCSCRIPT system view" [*SQL Anywhere Server - SQL Reference*].
  - New system procedures convert progress values. See:
    - "sa_convert_ml_progress_to_timestamp system procedure" [*SQL Anywhere Server - SQL Reference*]
    - "sa_convert_timestamp_to_ml_progress system procedure" [*SQL Anywhere Server - SQL Reference*]

- **New scheduling options for dbmlsync** When using the EVERY and INFINITE scheduling options, you can now specify that a synchronization does not occur when dbmlsync starts.

  See "NoSyncOnStartup (nss) extended option" [*MobiLink - Client Administration*].

- **Download-only publications** You can now create publications that only download data. Download-only publications do not use a log file.

  See "Download-only publications" [*MobiLink - Client Administration*].

- **Error handling enhancements** New event hooks have been added that allow you to process errors reported by dbmlsync on the client.

See:

- "Handling errors and warnings in event hook procedures" [*MobiLink - Client Administration*]
- "sp_hook_dbmlsync_all_error" [*MobiLink - Client Administration*]
- "sp_hook_dbmlsync_communication_error" [*MobiLink - Client Administration*]
- "sp_hook_dbmlsync_misc_error" [*MobiLink - Client Administration*]
- "sp_hook_dbmlsync_sql_error" [*MobiLink - Client Administration*]

- **Stop dbmlsync from enforcing table order** By default, dbmlsync issues an error if a child table is uploaded before a parent table. A new extended option allows you to override this behavior.

  See "TableOrderChecking (toc) extended option" [*MobiLink - Client Administration*].

- **Persistent connections** You can now specify that dbmlsync should keep open the connection to the MobiLink server between synchronizations.

  See "-pc+ dbmlsync option" [*MobiLink - Client Administration*].

- **New way to track synchronizations** For SQL Anywhere remote databases only, you can now specify a subscription_id parameter in your begin_publication or end_publication script. This value is called sync_id in the SYSSYNC system table. This is an advanced feature that helps you track information about your synchronizations. See:

  - "begin_publication connection event" [*MobiLink - Server Administration*]
  - "end_publication connection event" [*MobiLink - Server Administration*]

- **dbmlsync now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the non-ambiguous ISO 8601 datetime format: `{I|W|E}.yyyy-mm-dd hh:mm:ss message`.

- **Expanded values in #hook_dict** The dbmlsync utility exposes hooks and passes values as name/value pairs through a temporary table called #hook_dict. In the past, the values in the #hook_dict table were defined as VARCHAR(255). This has been increased to VARCHAR(10240).

Security

- **RSA now included with SQL Anywhere** You no longer have to purchase a separate license to use RSA encryption, unless you are using FIPS.

- **ECC encryption available with HTTPS** You can now use ECC encryption when you use HTTPS.

- **New mlsrv10 -fips option** You can now specify -fips when you start the MobiLink server, and thus force all secure connections to use FIPS-approved algorithms. This setting does not affect nonsecure streams.

  See "-fips mlsrv12 option" [*MobiLink - Server Administration*].

- **New mluser -fips option** The MobiLink user authentication utility now provides an option that enforces the use of FIPS security.

See "MobiLink user authentication utility (mluser)" [*MobiLink - Server Administration*].

- **FIPS security is supported on more platforms** FIPS security is now supported on more platforms. For a list of supported platforms, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

- **Simplified way to specify security streams** The syntax for specifying security options has been simplified on both the server and the client by treating security options as separate network protocols. The following protocols are now supported: TCP/IP, TLS (synchronization over TCP/IP with TLS security), HTTP, and HTTPS. The UltraLite Security parameter is removed.

  See:

  - MobiLink server: "-x mlsrv12 option" [*MobiLink - Server Administration*]
  - MobiLink clients: "MobiLink client network protocol option summary" [*MobiLink - Client Administration*]

- **Integration with operating system** By default, MobiLink clients now trust certificates that are already trusted by the operating system on which they operate.

Server-initiated synchronization

Ease of use

- **Server-initiated synchronization is much easier to set up** Enhancements have been made to make it much quicker to set up a server-initiated synchronization application:

  - **Sybase Central support** Notifiers and Listeners can now be set up in Sybase Central Model mode, allowing a subset of useful Notification services. In Model mode, you identify a table for server-initiated synchronization, and your download_cursor is automatically used to determine what data is used for notification purposes. When data identified in your download cursor changes, a Notification is sent. The Deployment wizard generates a corresponding Listener options file.

    See "Setting up server-initiated synchronization in a synchronization model" [*MobiLink - Getting Started*].

  - **New default gateway** A new gateway called the SYNC gateway allows you to make a persistent connection over the same type of communication path you use for MobiLink synchronization. The SYNC gateway is now the default device tracker gateway, meaning that notification will first try the SYNC gateway, with fallback to the UDP and then SMTP gateways.

    See "Gateways and carriers" [*MobiLink - Server-Initiated Synchronization*].

Notifier enhancements

- **Shared connections** Multiple Notifiers can now share the same database connection, reducing contention and required server resources in the consolidated database.

  See "Notifier events" [*MobiLink - Server-Initiated Synchronization*].

- **Notifier uses character set of remote device** Notifications are now sent to the remote device using the character set of the remote device. Device tracking information is translated before being applied to the consolidated database.

- **Custom confirmation handling** You can now implement a Notifier property in SQL that processes the confirmation of a push request and returns its status.

  See "confirmation_handler event" [*MobiLink - Server-Initiated Synchronization*].

- **Custom error handling** You can now implement a Notifier property in SQL that processes errors such as when a push request is not delivered, not confirmed, or improperly confirmed.

  See "error_handler event" [*MobiLink - Server-Initiated Synchronization*].

Listener enhancements

- **Persistent connections** The Windows Listener now supports persistent connections. By default, the Listener now maintains a persistent connection to the MobiLink server for device tracking, notification, and confirmation. This feature provides significant performance enhancement over previous versions. It can be disabled with the dblsn -pc option.

  See "MobiLink Listener utility for Windows devices (dblsn)" [*MobiLink - Server-Initiated Synchronization*].

- **New or changed Windows Listener options** The Listener now supports the following options:

Option	Description
-ni	Stop tracking UDP addresses when -x is used. Previously, this was called -g.
-pc{+|-}	Enable/disable persistent connection for notifications.
-ns	Disables default SMS listening on Windows Mobile 2003 and later Phone Edition.
-nu	Disable default UDP listening.
-r	Register the remote ID file for use by the $remote_id variable.
-v	When set to 1 or above, the verbosity option now displays and logs command line options.

- **Remote ID file** On the Listener command line, you can now access the new MobiLink remote ID (which by default is a GUID) using a remote ID file. You do this with the new dblsn option -r and new Listener action variable $remote_id.

See "MobiLink Listener utility for Windows devices (dblsn)" [*MobiLink - Server-Initiated Synchronization*] and "MobiLink Listener action commands for Windows devices" [*MobiLink - Server-Initiated Synchronization*].

- **New Listener action variables for authentication** There are new action variables that are useful in message handlers: $ml_user and $ml_password.

  See "MobiLink Listener action commands for Windows devices" [*MobiLink - Server-Initiated Synchronization*].

- **New Listener action variable for connection parameters** The new $ml_connect action variable expands to the MobiLink connection parameters that were specified with the dblsn -x option.

  See "MobiLink Listener action commands for Windows devices" [*MobiLink - Server-Initiated Synchronization*].

- **Listener now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the unambiguous ISO 8601 datetime format: **{I|W|E}**.*yyyy-mm-dd hh:mm:ss message*.

- **Listener can use TLS** The Listener can now connect to the MobiLink server using all the network choices as other MobiLink clients. This allows you to apply security to device tracking and notification.

  See -x in "MobiLink Listener utility for Windows devices (dblsn)" [*MobiLink - Server-Initiated Synchronization*].

Increased device support

- **Support for Treo 600 and 650** The Palm Listener now supports Treo 600 and 650.
- **CE Phone Edition support** The Listener now supports Windows Mobile 2003 Phone Edition for SMS.

Behavior changes and deprecated features

Following is a list of changes to MobiLink introduced in version 10.0.0.

MobiLink server changes

Changes to MobiLink scripts

- **Cursor-based uploads removed** The following scripts were deprecated in version 9.0.0 and are now removed: upload_cursor, new_row_cursor, and old_row_cursor. You should instead use statement-based scripts.

  See "Writing scripts to upload rows" [*MobiLink - Server Administration*].

- **Unrecognized scripts cause the synchronization to fail** If the MobiLink server encounters any unrecognized table-level or connection-level scripts, it will abort the synchronization. In previous

versions, unrecognized scripts only resulted in a warning message. This means that the presence of cursor-based upload scripts cause the synchronization to abort.

- **Errors in upload or download scripts cause the synchronization to fail** The synchronization now always aborts if the MobiLink server encounters errors with upload or download scripts. Previously, the MobiLink server did not always abort the synchronization.

- **The handle_error and handle_odbc_error events work in a more restricted fashion** The handle_error and handle_odbc_error scripts are now only called when an ODBC error occurs while MobiLink is processing an insert, update, or delete script during the upload transaction, or is fetching download rows. If an ODBC error occurs at another time, the MobiLink server will call the report_error or report_odbc_error script and abort the synchronization.

- **Authentication scripts committed** If there is no error, the MobiLink server always commits the transaction after invoking an authenticate_user, authenticate_user_hashed, or authenticate_parameters script, even if the authentication fails. Previously, transactions involving failed authentication were rolled back, so there could be no record of failed attempts to authenticate. See:

  - "authenticate_user connection event" [*MobiLink - Server Administration*]
  - "authenticate_user_hashed connection event" [*MobiLink - Server Administration*]
  - "authenticate_parameters connection event" [*MobiLink - Server Administration*]

- **Changes to authenticate_user_hashed script** The authenticate_user_hashed script can now be called more than once during an authentication sequence for a user.

  See "authenticate_user_hashed connection event" [*MobiLink - Server Administration*].

- **When a begin script is called, its end script is called regardless of the success of the synchronization** There are several MobiLink scripts that have a begin and end form, such as begin_connection and end_connection. In the past, the end script was often not executed if the synchronization failed. Now, if the begin script is called, the end script is always called (if it is defined), even if the synchronization has errors.

  See "Synchronization events" [*MobiLink - Server Administration*].

- **Upload scripts are not called for a table when there is no data to upload** In previous versions, you could use the -us option to prevent the MobiLink server from calling upload scripts when there is no data to upload. The -us option is now removed and by default, the MobiLink server only invokes upload scripts when the upload stream contains data to upload. You can revert to the old behavior using the -zus option.

  See "-zus mlsrv12 option" [*MobiLink - Server Administration*].

- **SQL Anywhere 10 and Microsoft SQL Server 2005 consolidated databases should not change the isolation level in the begin_connection script** For SQL Anywhere version 10 and Microsoft SQL Server 2005 and later, the default isolation level for downloads is now snapshot. This means that the isolation level may be changed at the beginning of the download transaction, in which case any setting from the begin_connection script is overridden. Therefore, you should change the isolation level for downloads in the begin_download script or use the new mlsrv10 -dsd option to disable snapshot isolation. Previous documentation recommended changing the isolation level in the

begin_connection script, and this is still good practice for consolidated databases that do not use snapshot isolation.

See "MobiLink isolation levels" [*MobiLink - Server Administration*].

- **example_upload_cursor, example_upload_delete, example_upload_insert, and example_upload_update table events are removed** As a result of removing the -za and -ze MobiLink server options, the example_upload_cursor, example_upload_delete, example_upload_insert, and example_upload_update table events are no longer generated. You can now generate scripts using the Create Synchronization Model wizard.

  See "Creating synchronization models" [*MobiLink - Getting Started*].

mlsrv10 changes

- **-w and -wu options have changed** The -w and -wu options set the number of database worker threads and maximum number of uploading database worker threads, respectively. In previous versions, these worker threads performed every aspect of synchronization, including reading and writing to the network, unpacking and packing protocol bytes and row data, running scripts, and updating and fetching rows in the consolidated database.

  Now the worker threads affected by -w and -wu are database worker threads. These database worker threads are solely responsible for all database activity, and nothing more. Other threads are responsible for network activity, packing and unpacking, and all other MobiLink server activities.

  The new behavior of the -w and -wu options is independent of network activity. In previous versions, networks with high latency could cause worker threads to block, requiring some deployments to specify a large number of worker threads. The new MobiLink architecture removes this requirement.

  The -w and -wu options are still the simplest way to limit the load that the MobiLink server puts on the consolidated database. Testing -w and -wu values can help you find the best throughput for your synchronization system. See:

  - "-w mlsrv12 option" [*MobiLink - Server Administration*]
  - "-wu mlsrv12 option" [*MobiLink - Server Administration*]
  - "MobiLink performance" [*MobiLink - Server Administration*]

- **Options for setting cache size are removed** The following mlsrv10 options have been removed:

  - -bc
  - -d
  - -dd
  - -u

  These options have been replaced with the mlsrv10 -cm option, which sets the cache for all synchronizations.

  See "-cm mlsrv12 option" [*MobiLink - Server Administration*].

- **Options for setting timeout are removed** The following mlsrv10 options are no longer required and have been removed:

  - contd_timeout
  - unknown_timeout

  Them mlsrv10 liveness_timeout option has also been removed. It is replaced by the timeout option for synchronization clients.

  See "timeout" [*MobiLink - Client Administration*].

- **Backlog option no longer required** The mlsrv10 backlog option is no longer required and has been removed.

- **Changes to protocol names and options for network security** The following network protocol keywords have been removed: https_fips, rsa_tls, rsa_tls_fips, ecc_tls; as well as the network protocol option security. The protocols are not removed, but you specify them differently now. The mlsrv10 -x syntax has changed as follows:

Old syntax	New syntax in version 10.0.0	Description
-x https_fips	-x https(fips=y;...)	HTTPS with FIPS
-x rsa_tls	-x tls(tls_type=rsa;...)	TCP/IP with TLS using RSA encryption
-x rsa_tls_fips	-x tls(tls_type=rsa;fips=y;...)	TCP/IP with TLS using RSA encryption and FIPS
-x ecc_tls	-x tls(tls_type=ecc;...)	TCP/IP with TLS using ECC encryption
-x tcpip(security=...)	-x tcpip	TCP/IP
-x http(security=...)	-x http	HTTP

  See "-x mlsrv12 option" [*MobiLink - Server Administration*].

- **Change to -bn option** The mlsrv10 -bn option compares BLOB bytes during conflict detection. Previously, characters were compared for data of type LONGVARCHAR. Now the units that are compared are always bytes for both binary and LONGVARCHAR BLOBs.

  See "-bn mlsrv12 option" [*MobiLink - Server Administration*].

- **Change to verbosity output** The mlsrv10 options -vr, -vt, and -vu all output slightly different information:

  - **-vr** Now, -vr returns only the upload and download row values. Previously, the upload and download script names and contents were also returned.

- **-vt** Now, -vt returns only the contents of translated scripts. Previously, the original script contents were also returned.

- **-vu** Now, -vu returns all undefined table scripts when the scripts need to be invoked. This includes statistical scripts.

See "-v mlsrv12 option" [*MobiLink - Server Administration*].

- **MobiLink server options -za and -ze are removed** Automatic script generation provided by the MobiLink server -za option and -ze options has been removed. You can now generate scripts using the Create Synchronization Model wizard.

  See "Creating synchronization models" [*MobiLink - Getting Started*].

- **-zac and -zec are removed** The MobiLink server options for generating cursor-based scripts, -zac and -zec, were deprecated and are now removed.

- **MobiLink server option -oy is removed** The mlsrv10 -oy option, which showed the year in timestamps, has been removed. The year is now always shown in timestamps in informational, warning, and error messages.

Statistical properties

- **Changes to statistical properties** There are two general changes:

  - The meaning of byte counts for upload and download have changed. The counts now reflect the amount of memory used within the MobiLink server to store the upload and download. Previously, they were the size in bytes of the upload and download as sent to or received from the MobiLink server. The new counts are more useful because they provide a good indication of a synchronization's impact on server memory. Also, the previous counts could be unreliable when HTTP, encryption, or compression were used.

  - In previous versions of the documentation, the property descriptions did not explain that the properties return different values based on whether you are using them in normal upload mode or in forced conflict mode. This has been corrected (see below).

  The following statistical properties have changed:

Statistical property	Description
conflicted_deletes	In normal upload mode, this is always zero. In forced conflict mode, it returns the total number of uploaded deletes that were successfully inserted into the consolidated database using the upload_old_row_insert script. Previously, this returned the number of uploaded deletes for which conflicts were detected.

Statistical property	Description
conflicted_inserts	In normal upload mode, this is always zero. In forced conflict mode, it returns the total number of upload inserts that were successfully inserted into the consolidated database using the upload_new_row_insert script. Previously, this returned the number of uploaded inserts for which conflicts were detected.
conflicted_updates	In normal upload mode, this returns the total number of update rows that caused a conflict. In forced conflict mode, it returns the total number of upload update rows that were successfully applied using upload_new_row_insert or upload_old_row_insert scripts. Previously, this returned the number of uploaded updates for which conflicts were detected.
download_bytes	This returns the amount of memory used within the MobiLink server to store the download. Previously, this returned the number of downloaded bytes.
ignored_deletes	In normal upload mode, this returns the total number of upload delete rows that caused errors while the upload_delete script was invoked, when the handle_error or handle_odbc_error are defined and returned 1000, or when there is no upload_delete script defined for the given table. In forced conflict mode, this returns the total number of upload delete rows that caused errors while the upload_old_row_insert script was invoked, when the handle_error or handle_odbc_error are defined and returned 1000, or when there is no upload_old_row_insert script defined for the given table. Previously, this returned the number of uploaded deletes that were ignored.
ignored_inserts	In normal upload mode, this returns the total number of upload insert rows that caused errors while the upload_insert script was invoked, when the handle_error or handle_odbc_error are defined and returned 1000, or when there is no upload_insert script defined for the given table. In forced conflict mode, this returns the total number of upload insert rows that caused errors while the upload_new_row_insert script was invoked, when the handle_error or handle_odbc_error are defined and returned 1000, or when there is no upload_insert script defined for the given table. Previously, this returned the number of uploaded inserts that were ignored.

Statistical property	Description
ignored_updates	In normal upload mode, this returns the total number of upload update rows that caused errors while the upload_update script was invoked, when the handle_error or handle_odbc_error are defined and returned 1000, or when there is no upload_update script defined for the given table. In forced conflict mode, this returns the total number of upload update rows that caused errors while the upload_new_row_insert or upload_old_row_insert scripts were invoked, or when the handle_error or handle_odbc_error are defined and returned 1000. Previously, this returned the number of uploaded updates that were ignored.
upload_bytes	This returns the amount of memory used within the MobiLink server to store the upload. Previously, this returned the number of uploaded bytes.
upload_deleted_rows	In normal upload mode, this returns the total number of rows that were successfully deleted from the consolidated database. In forced conflict mode, this is always zero. Previously, this returned the number of row deletions that were uploaded from the synchronization client.
upload_inserted_rows	In normal upload mode, this returns the total number of rows that were successfully inserted in the consolidated database. In forced conflict mode, this is always zero. Previously, this returned the number of row insertions that were uploaded from the synchronization client.
upload_updated_rows	In normal upload mode, this returns the total number of rows that were successfully updated in the consolidated database. In forced conflict mode, this is always zero. Previously, this returned the number of row updates that were uploaded from the synchronization client.

See "MobiLink statistical properties" [*MobiLink - Server Administration*].

Other MobiLink server changes

- **Null characters can now be synchronized to and from columns with CHAR or NCHAR data types in the remote database** Previously in MobiLink, VARCHAR and CHAR column values containing null characters could cause a synchronization to fail. Now you can synchronize null

characters in remote database columns of data type CHAR, VARCHAR, LONG VARCHAR, NCHAR, NVARCHAR, AND LONG NVARCHAR.

- **New format for logging information, warning, and error messages** Previously, the MobiLink server logged messages in the following format:

  ```
  T.mm/dd hh:mm:ss. thread_id User_name: message
  ```

  Now, the MobiLink server logs messages in the following format:

  ```
  T. yyyy-mm-dd hh:mm:ss. synchronization_id: message
  ```

  For each synchronization, the first message in the log shows the remote ID, user name, script version, and client name (UltraLite or SQL Anywhere).

  The new format reduces the size of the output log without reducing the information that is provided.

- **New data type in system procedures for Oracle** In MobiLink system procedures that are used to register scripts, the script contents parameter now uses the CLOB data type for Oracle consolidated databases. In the ml_add_property system procedure, the prop_value parameter is now CLOB for Oracle. Previously, these parameters were type VARCHAR.

MobiLink user names and remote IDs

MobiLink now generates a unique ID called a remote ID the first time a remote database synchronizes (or when it encounters a NULL value for the remote ID). The MobiLink user name no longer needs to be unique. The MobiLink user name can now be considered a true user name that is used for authentication.

In previous versions, the synchronization progress was stored for the MobiLink user name. Now, the progress is stored for the remote ID and subscription for SQL Anywhere remotes, and the remote ID and publication for UltraLite remotes.

Previously, you used the MobiLink user name to uniquely identify a remote database. The remote ID is a useful way to identify the remote database when you want a MobiLink user to synchronize multiple remote databases. In UltraLite remote databases, the remote ID is also useful for multiple MobiLink users to synchronize the same remote database.

See “Remote IDs” [*MobiLink - Client Administration*].

UltraLite clients

See “Behavior changes and deprecated features” on page 290.

SQL Anywhere clients

- **Download error hooks deprecated** The following error hooks are deprecated: sp_hook_dbmlsync_download_com_error, sp_hook_dbmlsync_fatal_sql_error, and sp_hook_dbmlsync_sql_error. They have been replaced.

  See “Handling errors and warnings in event hook procedures” [*MobiLink - Client Administration*].

- **sp_hook_dbmlsync_log_rescan only called if dbmlsync expects another synchronization** Previously, the sp_hook_dbmlsync_log_rescan hook was called at the end of every synchronization. This caused a pause to occur after dbmlsync disconnected from the MobiLink server, but before the "synchronization complete" message was displayed in the log. Now, the hook is only called when dbmlsync expects another synchronization, for example when the dbmlsync -n option is specified more than once in a command line or when scheduling is enabled.

  See "sp_hook_dbmlsync_log_rescan" [*MobiLink - Client Administration*].

- **Liveness timeout options simplified** On the client, the liveness_timeout and network_connect_timeout network connection protocol options are removed. Use the timeout connection option instead.

  See "timeout" [*MobiLink - Client Administration*].

- **No compression means no obfuscation** If you set compression to none, data is now completely unobfuscated. If security is an issue, you should use transport-layer security to encrypt your data.

  See "compression" [*MobiLink - Client Administration*].

- **Version 7 syntax and utilities are removed** The following SQL statements and utility were deprecated and are now removed:

  - MobiLink client database extraction utility (mlxtract)
  - CREATE SYNCHRONIZATION SITE statement
  - CREATE SYNCHRONIZATION DEFINITION statement
  - CREATE SYNCHRONIZATION TEMPLATE statement

- **New network protocol options for ActiveSync** ActiveSync users no longer have to specify the ActiveSync protocol when specifying the CommunicationAddress extended option or the ADDRESS clause in SQL statements. Instead, for ActiveSync you just specify the protocol and protocol options you are using for communication between the MobiLink provider for ActiveSync and the MobiLink server.

  See "MobiLink client network protocol option summary" [*MobiLink - Client Administration*].

- **New way to shut down dbmlsync** The dbmlsync -k option is deprecated and replaced with the -qc option.

  See "-qc dbmlsync option" [*MobiLink - Client Administration*].

Miscellaneous MobiLink behavior changes

Version support

- **Support for clients before version 8.0.0 removed** The MobiLink server no longer supports SQL Anywhere clients before version 8.0.0. To use older databases with the version 10 MobiLink server, you need to follow upgrade procedures.

See "Upgrading SQL Anywhere" on page 304.

Name changes

The following utility names have changed:

Old utility name	New utility name
dbmlsrv9	mlsrv10
dbmluser	mluser
dbmlmon	mlmon
dbmlstop	mlstop
dbasinst	mlasinst

The following file names have changed:

Old file name	New file name
dbmlsv9.dll	*mlodbc10.dll*
dbasdesk.dll	*mlasdesk.dll*
dbasdev.dll	*mlasdev.dll*
dbmlsrv.mle	*mlsrv10.mle*
syncasa.sql	*syncsa.sql*

ODBC driver enhancements

MobiLink now uses new drivers for Adaptive Server Enterprise and DB2 consolidated databases.

See: "Changes to ODBC drivers used by MobiLink, QAnywhere, and remote data access" on page 301.

Server-initiated synchronization

- **Windows SDK removed** The SDK for creating support for more Windows devices has been removed. It is replaced by improved support for SMS. The Palm SDK remains.

- **Listener -g option is replaced** The dblsn -g option is replaced with the dblsn -ni option.

Other MobiLink behavior changes

- **Support for Windows Performance Monitor is dropped** MobiLink no longer supports Windows Performance Monitor. You should use the MobiLink Monitor instead.

  See "MobiLink Monitor" [*MobiLink - Server Administration*].

- **Server-initiated synchronization no longer supports Kyocera devices** There is no longer a Palm SDK for Kyocera devices. The Palm Listener continues to support Treo devices.

QAnywhere

The following sections describe the new features, behavior changes, and deprecated features in QAnywhere for version 10.0.0.

New features

Following is a list of additions to QAnywhere introduced in version 10.0.0.

Mobile web services

Mobile web services provide support for mobile-optimized asynchronous web services. This allows mobile applications to make web service requests—even when they are offline—and have those requests queued for transmission later. The requests are delivered as messages using QAnywhere. A web services connector on the server side takes the client request and forwards it to the web service. It then takes the response from the web service and returns it to the client as a message. The provided WSDL compiler facilitates the use of mobile web services from your .NET or Java application.

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

New QAnywhere plug-in for Sybase Central

Sybase Central now includes a QAnywhere plug-in that provides an easy-to-use graphical interface for creating and administering your QAnywhere applications. With the QAnywhere plug-in, you can:

- Create client and server message stores.
- Create and maintain configuration files for the QAnywhere Agent.
- Browse QAnywhere Agent log files.
- Create or modify destination aliases.
- Create JMS connectors and web service connectors.
- Create and maintain transmission rules files.
- Browse message stores remotely.
- Track messages.

Although QAnywhere is not supported on Unix platforms, you can now use Sybase Central on Unix to track messages.

New QAnywhere client APIs

- **New SQL API** The QAnywhere SQL API is a set of SQL stored procedures that allow SQL developers to easily leverage QAnywhere messaging capabilities. Using this API, stored procedures can send or receive messages using a straightforward approach that complements existing database applications. This can allow for powerful applications that combine database and messaging operations in a single transaction. For example, a stored procedure could insert a row into the database and send a message to another application—and have both actions committed as part of the same transaction.

  See "QAnywhere SQL API reference" [*QAnywhere*].

- **New Java client API** The new QAnywhere client API for Java helps you create messaging client applications in Java. The client API for Java is currently supported on Windows, including Windows CE.

  See "QAnywhere Java API reference for clients" [*QAnywhere*].

QAnywhere client API enhancements

The following additions have been made to the QAnywhere client APIs:

- **Message selectors** You can now use SQL-like expressions to selectively browse or receive messages from a queue. The syntax for creating message selectors is identical to that used for conditions in transmission rules.

  See "Browsing QAnywhere messages" [*QAnywhere*].

- **New ways to browse messages** You can now browse messages from multiple queues, or browse subsets of messages based on ID or message selector.

  See "Browsing messages using a selector" [*QAnywhere*].

- **Enumerate message store property names** You can now enumerate message store property names.

  See "Enumerating client message store properties" [*QAnywhere*].

- **Undeliverable messages** Using the new message store property ias_MaxDeliveryAttempts, you can set the maximum number of attempts that a QAnywhere client will attempt to receive a message before considering it undeliverable.

  See "Rule variables" [*QAnywhere*].

- **Canceling messages** You can now cancel messages before they are sent.

  See "Canceling QAnywhere messages" [*QAnywhere*].

- **Query message status** You can now query the status of a message using new pre-defined message properties: ias_Status and ias_StatusTime. You can also query the originator of a message with ias_Originator, or the number of times the message has been delivered to a receiver with ias_DeliveryCount.

See "Predefined message properties" [*QAnywhere*].

- **New message store property to set upload increments** ias_MaxUploadSize can be used to change the upload increment.

  See "Predefined client message store properties" [*QAnywhere*].

QAnywhere Agent new features

- **Multiple agents on a single device** Previously, you could only run one instance of the QAnywhere Agent on a device. This limitation has been removed.

  See "Starting the QAnywhere agent" [*QAnywhere*].

- **More options for setting up failover** There are two new QAnywhere Agent options, -fd and -fr, that help you customize the way failover occurs.

  See "-fd qaagent option" [*QAnywhere*] and "-fr qaagent option" [*QAnywhere*].

- **Persistent connections** The new option -pc+ has been added to enable persistent connections for message transmission. The new -push option replaces -push_notifications and now allows you to specify whether you want push notifications to use persistent connections.

  See:

  - "-pc qaagent option" [*QAnywhere*]
  - "-push qaagent option" [*QAnywhere*]

- **New upgrade procedure** The new -sur option can be used to upgrade a client message store from a previous version of SQL Anywhere.

  See "-sur qaagent option" [*QAnywhere*].

- **QAnywhere Agent now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the non-ambiguous ISO 8601 datetime format: `{I|W|E} yyyy-mm-dd hh:mm:ss message`.

Other QAnywhere enhancements

- **Destination aliases** You can now define a destination alias that represents a set of QAnywhere destinations. Messages sent to a destination alias are sent to each member of the alias.

- **Server management requests** You can now use server management requests for administration and monitoring activities such as creating destination aliases or monitoring, starting, and stopping JMS connectors. You create server management requests on the client and send them to the server message store for processing.

  See "Server management requests" [*QAnywhere*].

- **Improved maintenance of server transmission rules** You can now change the default server transmission rules and the change will automatically be applied to all clients. Previously, to change the default you had to manually define a transmission rule for each client.

  See "Server transmission rules" [*QAnywhere*].

- **More message properties** Additional pre-defined message properties are set by QAnywhere, giving you more flexibility in processing messages, better information during debugging, and more help with troubleshooting the status of messages.

  See "Message properties" [*QAnywhere*].

- **Ability to embed backslashes in JMS destinations** JMS destinations can now include subcontexts that require backslash delimiters.

  See "Sending a QAnywhere message to a JMS connector" [*QAnywhere*].

- **New transmission rule functions** The following transmission rule functions have been added for improved date handling:

  - DATEADD(datepart, count, datetime)
  - DATEPART(datepart, date)
  - DATETIME(string)
  - LENGTH(string)
  - SUBSTR(string, start, length)

  See "Rule functions" [*QAnywhere*].

- **Prefaces for properties in transmission rules** You can now preface message property names and message store property names when you use them in transmission rules and so bypass the precedence given to transmission rule variables of the same name.

  See "Using properties as rule variables" [*QAnywhere*].

Behavior changes and deprecated features

Following is a list of changes to QAnywhere introduced in version 10.0.0.

QAnywhere client changes

- **Client message store ID has changed** The client message store ID is now a MobiLink remote ID. Previously it was a MobiLink user name. You do not have to register a remote ID with the consolidated database. However, you still need to register a MobiLink user name with the server message store. If you do not specify a MobiLink user name, it defaults to the client message store ID.

New qaagent options have been provided to manage your MobiLink user names. See:

- "-mn qaagent option" [*QAnywhere*]
- "-mp qaagent option" [*QAnywhere*]
- "-mu qaagent option" [*QAnywhere*]

- **New ODBC drivers** The iAnywhere Solutions ODBC drivers for connecting to Adaptive Server Enterprise and DB2 server message stores have changed.

  See "Changes to ODBC drivers used by MobiLink, QAnywhere, and remote data access" on page 301.

QAnywhere Agent changes

- **qaagent -port is removed** The -port option specified a port number on which QAnywhere Agent listened for communications from the Listener. This option is no longer required and has been removed. A free port is automatically used.

- **qaagent -la_port is replaced** The -la_port option has been replaced by the -lp option.

  See "-lp qaagent option" [*QAnywhere*].

- **qaagent -push_notifications is renamed** This option is now called -push. It now allows you to enable push notifications with or without persistent connection.

  See "-push qaagent option" [*QAnywhere*].

- **Changes to policy defaults** The default policy is now automatic. Previously it was scheduled. The default schedule interval is now 900 seconds (15 minutes). Previously it was 10 seconds.

  See "-policy qaagent option" [*QAnywhere*].

- **Transaction log is not used or maintained** The QAnywhere Agent no longer uses a transaction log or manages its size. As a result, for most applications the client message store should be created using the dbinit -n option, which initializes the database with no transaction log.

  See "Setting up the client message store" [*QAnywhere*].

Other QAnywhere changes

- **Server-side property files are deprecated** Instead of storing properties in files, you now store them in the database.

- **getPropertyNames** The getPropertyNames function has been removed from the C++ client API. It has been replaced with beginEnumPropertyNames, nextPropertyName, and endEnumPropertyNames.

  See "QAMessage class" [*QAnywhere*].

- **Date handling in transmission rules** The following transmission rule message store variables have been removed:

  - ias_CurrentDayOfWeek
  - ias_CurrentDayOfMonth
  - ias_CurrentMonth
  - ias_CurrentYear

  In their place, you can use ias_CurrentTimestamp or DATEPART.

  See "Rule variables" [*QAnywhere*].

- **QAnywhere Central replaced** QAnywhere Central has been replaced with the QAnywhere plug-in to Sybase Central. The plug-in provides many enhancements in functionality.

SQL Remote

The following sections describe the new features, behavior changes, and deprecated features in SQL Remote for version 10.0.0.

New features

Following is a list of additions to SQL Remote introduced in version 10.0.0.

- **invalid_extensions option** A new messaging option has been added that allows you to stop SQL Remote from using certain file extensions in FILE and FTP messaging.

  See "SET REMOTE OPTION statement [SQL Remote]" [*SQL Anywhere Server - SQL Reference*].

- **The Message Agent (dbremote) has a graphical user interface on Unix and Linux platforms** See -ux option in "SQL Remote Message Agent utility (dbremote)" [*SQL Remote*].

- **dbremote now uses ISO 8601 datetime format for message timestamps** Timestamps in informational, warning, and error messages now use the non-ambiguous ISO 8601 datetime format: `{I|W|E} yyyy-mm-dd hh:mm:ss message`.

- **New option for dbremote** To close window on completion, use the new -qc option.

  See "SQL Remote Message Agent utility (dbremote)" [*SQL Remote*].

Behavior changes and deprecated features

Following is a list of changes to SQL Remote introduced in version 10.0.0.

- **Support for Adaptive Server Enterprise databases is removed** SQL Remote no longer supports Adaptive Server Enterprise consolidated databases. This means that ssxtract, ssremote,

ssqueue, and all other SQL Remote for Adaptive Server Enterprise utilities and files are removed from the install.

To synchronize Adaptive Server Enterprise databases, you should use MobiLink.

For information about upgrading from SQL Remote to MobiLink, see http://www.ianywhere.com/whitepapers/migrate_to_ml.html.

- **Extract Database wizard no longer extracts legacy databases** The Extract Database wizard only works with version 10 databases.
- **Expanded values in #hook_dict** The utilities dbxtract and dbremote expose hooks and pass values as name/value pairs through a temporary table called #hook_dict. In the past, the values in the #hook_dict table were defined as VARCHAR(255). This has been increased to VARCHAR(10240).
- **Changes to Extraction utility** There are several changes to dbxtract:
  - The options -j, -k, and -x are removed.
  - New options -al and-xh are added.

  See "Extraction utility (dbxtract)" [*SQL Remote*].
- **Message Agent (dbremote) deprecated option** The -k option, which closed the window on completion, is deprecated. To close window on completion, use the new -qc option.

UltraLite

The following sections describe the new features, behavior changes, and deprecated features in UltraLite for version 10.0.0.

New features

Following is a list of additions to UltraLite introduced in version 10.0.0.

Main features

UltraLite is now a full-featured relational database management system, designed with ease-of-administration and SQL Anywhere compatibility in mind. Despite the addition of many new and useful features, UltraLite still maintains a small footprint size. For a complete list of UltraLite limitations for this release, see "UltraLite limitations" [*UltraLite - Database Management and Reference*].

Main features of this release include:

- **Increased database limits** The UltraLite database limits have been dramatically increased. In particular, the maximum number of rows in a table has been increased to 16 million. For other current database limits, see "UltraLite limitations" [*UltraLite - Database Management and Reference*].

- **Integrated schema** UltraLite is now a standalone RDBMS and no longer requires a separate schema file to define the logical structure of the database. For this release, the UltraLite schema is fully integrated with the database. See "UltraLite database schema" [*UltraLite - Database Management and Reference*] for details on the internal database schema.

- **Consolidated file formats** File formats have been consolidated in version 10 of UltraLite. This means that most platforms can now share a database file. If you need characters that are not defined by the collation you require, you should now choose to UTF-8 encode your database. See "UltraLite platform requirements for character set encoding" [*UltraLite - Database Management and Reference*] and "UltraLite utf8_encoding creation parameter" [*UltraLite - Database Management and Reference*] for details.

- **Increased database performance and data integrity** Overall, the UltraLite database performance and data integrity has been improved with several indexing and database page management improvements.

- **Indexes may utilize hashing** Indexes may now be specified to utilize hashing. The hash size can be specified on a per-index basis. The hash size can improve performance of index lookups and may affect database file size. See "UltraLite performance and optimization" [*UltraLite - Database Management and Reference*].

- **Direct database creation** You can now create UltraLite database files directly; a database schema file or reference database file is not required as the source for an UltraLite database. Instead, you can independently create an UltraLite database with Sybase Central or a command line utility, or even programmatically from an application.

  For existing UltraLite users, you can no longer create databases in the same manner as previous versions. See "Upgrading UltraLite" on page 329.

- **Direct Windows CE support** With this release, UltraLite applications from the desktop can connect directly to databases deployed to a Windows CE device. You can specify an UltraLite database by specifying the path and name and prefix it with **WCE:**. These direct access is supported by all client applications and administration tools, including Sybase Central and Interactive SQL. See "Windows Mobile" [*UltraLite - Database Management and Reference*].

- **Embedded SQL as a dynamic SQL programming interface** In previous versions, embedded SQL was a static interface. In this version, it is an interface to UltraLite dynamic SQL and does not require a SQL Anywhere database. Embedded SQL support also supports dynamic ESQL statements and the use of host variable placeholders. Furthermore, ESQL applications can also now run with uleng10. You can achieve this by linking against *ulrtc.lib* instead of *ulrt.lib*.

  As a result of this change, you may notice that simple embedded SQL applications could grow in size, whereas complex applications may become smaller. See "Upgrading UltraLite" on page 329 and "Developing embedded SQL applications" [*UltraLite - C and C++ Programming*].

Platforms and devices

Platform support has been modified. For a list of supported platforms, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

Important enhancements to note include:

- **Deployment Platforms** Platform enhancements include:
  - **Palm OS** UltraLite support of Palm OS devices has been enhanced with the following changes:
    - Runtime Support for Palm OS v4.x and later.
    - Development support for CodeWarrior has been increased to version 9. Note that CodeWarrior 8 is no longer supported.
    - Support for multiple databases and generalized file names. You can now specify a database for multiple devices with the DBF connection parameter, ensuring that you set the file name correctly depending on whether you are using record-based or file-based storage. See "UltraLite DBF connection parameter" [*UltraLite - Database Management and Reference*] and "Specifying file paths in an UltraLite connection parameter" [*UltraLite - Database Management and Reference*].
    - Support for both NVFS devices and VFS devices.
  - **Symbian OS** Symbian OS support is new to this version of UltraLite. UltraLite supports versions 7.0 and 8.0 of Symbian OS on UIQ phones (2.0 and 2.1) and Nokia S60 (second edition), and Series 80 devices.
  - **Windows Mobile 2005** If you are using Embedded Visual C++ 3.0 or 4.0, you can continue to use the existing runtimes. However, new runtimes (installed under *\ultralite\ce\arm.50*) are required when using Visual Studio 2005 to build an application.
- **Enhanced development environment support** Development tools and languages have been updated as follows:
  - UltraLite now supports ADO.NET 1.0 development in Visual Studio .NET 2003 and ADO.NET 2.0 development in Visual Studio 2005.
  - UltraLite now supports AppForge Crossfire version 5.6 for Visual Basic and C# development. You can deploy applications for AppForge to Palm OS, Symbian OS, and Windows CE platforms.
  - C++ component development.

Security

- **Encryption types** If you are compressing over TLS, UltraLite now supports both ECC and RSA encryption. RSA encryption is no longer separate product. See "Configuring UltraLite clients to use transport-layer security" [*SQL Anywhere Server - Database Administration*].

- **FIPS security** You can now secure MobiLink server communications with FIPS security.

- **Simplified security streams** You can now define encrypted streams as a network protocol or stream type rather than use a separate security parameter. The complete set of supported stream types is: TCP/IP, TLS (for RSA, ECC, and FIPS), HTTP, and HTTPS. See "Stream Type synchronization parameter" [*UltraLite - Database Management and Reference*].

Database management

Important new features and enhancements include:

- **Password changes** All passwords are case sensitive, regardless of the case-sensitivity of the database. New databases are created with a default user ID of **DBA** with the password **sql**. Consequently user IDs, passwords and trusted root certificates are not preserved as you upgrade your database from earlier releases.

- **Improved database properties and connection parameters** Database properties and connection parameters have been enhanced and simplified to allow you describe database and connection behavior more easily. See "UltraLite database properties" [*UltraLite - Database Management and Reference*] and "UltraLite connection parameters" [*UltraLite - Database Management and Reference*] for a complete list of database properties and connection parameters you can set for this release of the database.

- **Increased index performance** UltraLite index performance has been enhanced with this release. One of the major improvements is the introduction of index hashing. UltraLite now has a configurable index hash size. By setting the hash size to a value between 1-32 bytes, UltraLite stores part or all of the indexed value in the index page. This reduces the number of row lookups required. See "UltraLite max_hash_size creation parameter" [*UltraLite - Database Management and Reference*].

- **Checksum validation** You can now include checksums on database pages to validate data integrity of these pages as they are stored on disk. See "UltraLite checksum_level creation parameter" [*UltraLite - Database Management and Reference*].

- **Extended BLOB support** UltraLite databases now have extended support for BLOBs. UltraLite allows you to update, cast data types and get the length of these BLOBs.

Administration tools

Administration tools have been enhanced with this release. To ensure the correct usage of a tool, ensure you review the documentation for it.

Graphical administration tools

- **Sybase Central** You can now use Sybase Central to create, modify, and administer your UltraLite databases in a graphical user interface. This replaces the ulview schema editing utility.

  The list of wizards included in Sybase Central include:

  - Use the Create Database wizard to build a new UltraLite database. This wizard shares the same functionality as the ulcreate utility.
  - Use the Erase Database wizard to erase an existing UltraLite database. No utility equivalent exists for this wizard.
  - Use the Extract Database wizard to initialize a new UltraLite database from a SQL Anywhere reference database. This wizard shares the same functionality as the ulinit utility.
  - Use the Load Database wizard to load an XML file into an UltraLite database. This wizard shares the same functionality as the ulload utility.
  - Use the Migrate C++ API wizard to migrate C/C++ code created with the removed ulgen utility. No utility equivalent exists for this wizard.
  - Use the Synchronize Database wizard to synchronize an UltraLite database. This wizard shares the same functionality as the ulsync utility.
  - Use the Upgrade Database wizard to upgrade an existing UltraLite database from a previous version. This wizard shares the same functionality as the ulunloadold utility when used with the ulload utility.
  - Use the Unload Database to unload data/schema information from an UltraLite database to XML, SQL, or another database. This wizard shares the same functionality as the ulunload utility with the additional functionality of the ulcreate and ulload utilities.

- **Interactive SQL** You can now use Interactive SQL to develop and test SQL statements with UltraLite databases. Interactive SQL replaces the ulisql utility used in previous versions. See "Interactive SQL for UltraLite utility (dbisql)" [*UltraLite - Database Management and Reference*].

Command line administration tools

The following command line utilities are new to UltraLite:

- **Unload Old Database utility** The new ulunloadold command line utility helps you to unload existing 8.0.2 or 9.x UltraLite databases (schema + data) or schema files to an XML file. With the ulload command line utility, you can then use that output to rebuild an UltraLite version 10 database. See "UltraLite Unload Old Database utility (ulunloadold)" [*UltraLite - Database Management and Reference*].

- **Information utility** The new ulinfo utility displays information about an UltraLite database. It can also change and/or clear database option IDs like global_id or ml_remote_id. See "UltraLite Information utility (ulinfo)" [*UltraLite - Database Management and Reference*].

Also, because existing command line utilities have been enhanced to support the new RDBMS features in UltraLite 10, the options for these utilities have changed from earlier versions. To ensure you are using

new utilities correctly, ensure you review the reference documentation before starting. See "UltraLite utilities" [*UltraLite - Database Management and Reference*] for complete utility reference notes.

- **Enhanced error reporting** UltraLite utilities now report errors consistently with other SQL Anywhere utilities.
- **Extended database creation parameters** All database creation utilities (for example, ulcreate and ulload) now support the use of extended creation parameters. These extended creation parameters are configured on the command line with -o, and allow you to configure the same set of features that you can configure with Sybase Central wizards. See "Choosing database creation parameters for UltraLite" [*UltraLite - Database Management and Reference*].
- **Enhanced unload behavior** You can now use ulunload to output the UltraLite database schema as a sequence of dynamic SQL statements. See "UltraLite Database Unload utility (ulunload)" [*UltraLite - Database Management and Reference*].
- **Enhanced ulsync behavior** ulsync allows you to set network protocol options and extended synchronization parameters directly from this utility. See "UltraLite synchronization parameters and network protocol options" [*UltraLite - Database Management and Reference*] for a complete list.

  Additionally, ulsync now allows you to name publications, not just the publication mask. The keyword **Publications** takes a comma separated list of publication names. See "UltraLite Synchronization utility (ulsync)" [*UltraLite - Database Management and Reference*] for details.
- **Enhanced conduit installation** The HotSync Conduit Installation utility (ulcond10) now supports conduit extensions, connection strings, and multiple databases.
- **ulmvbreg** The ulmvbreg utility that registers UltraLite for AppForge has been renamed to ulafreg. This utility is now installed to the *install-dir\win32* directory.

ULSQLCONNECT

Previously, all UltraLite utilities received connection information from the command line. Now, if you want to pass information other than default user IDs and passwords, you can set the ULSQLCONNECT environment variable on your host machine. See "Storing UltraLite parameters with the ULSQLCONNECT environment variable" [*UltraLite - Database Management and Reference*].

SQL

- **SQL Statements** UltraLite supports several new statements. These new statements include:
  - **ALTER TABLE** In addition to creating tables with UltraLite SQL, you can now alter the definition with this statement. See "ALTER TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].
  - **ALTER/CREATE/DROP PUBLICATION** UltraLite now supports the addition, creation, and deletion of publications with these three statements. See "ALTER PUBLICATION statement [UltraLite][UltraLiteJ]" [*UltraLite - Database Management and Reference*],"CREATE PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and*

Reference] and "DROP PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **START/STOP SYNCHRONIZATION DELETE** UltraLite SQL includes these statements. Use these statements to control how deletes are logged with MobiLink synchronization. See "START SYNCHRONIZATION DELETE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*] and "STOP SYNCHRONIZATION DELETE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **Named constraints** Table constraints can now be named in the ALTER TABLE and CREATE TABLE statements. This permits modification of table and column constraints by changing individual constraints, rather than by modifying an entire table constraint. See "ALTER TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*] and "CREATE TABLE statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **Other SELECT statement enhancements** The SELECT statement has been extended:

  - SELECT statements can now include START AT as part of the TOP clause. START AT provides additional flexibility in queries that explicitly limit the result set. See "SELECT statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*].

  - The DISTINCT clause has been enhanced to allow aggregate functions with this clause (for example, SUM, AVERAGE, MAX, and so on). If you use aggregate functions with this clause, you will significantly increase the execution time. See "SELECT statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*] and "SQL functions" [*SQL Anywhere Server - SQL Reference*].

- **UNION operator** The UNION operator allows you to build a single result set from two or more queries into a single result set. By default, the UNION operator removes duplicate rows from the result set. See "Combining sets with the UNION clause" [*SQL Anywhere Server - SQL Usage*].

Synchronization

- **Configurable and increased default cache size for HotSync conduit synchronization** Previously, beyond a certain amount of data synchronized on Palm OS file-based data stores, synchronization speeds were negatively affected. Consequently, the default cache size (on desktop) for the UltraLite conduit has been increased to 4 MB. This increased cache size significantly improves the synchronization time by cutting down unnecessary file I/O operations. However, you can also configure a different default cache size if you choose.

- **Predicates on publications** Synchronization publications for UltraLite now allow predicates. If you want to optionally combine conditional expressions with the logical operators AND and OR, you can now define this set of conditions in a WHERE or HAVING clause. As with SQL Anywhere, a predicate that evaluates to UNKNOWN is interpreted as FALSE. "CREATE PUBLICATION statement [UltraLite] [UltraLiteJ]" [*UltraLite - Database Management and Reference*] and "ALTER PUBLICATION statement [UltraLite][UltraLiteJ]" [*UltraLite - Database Management and Reference*].

- **Improved MobiLink client network layer** Improvements to the client network layer include the following:
  - Synchronization compression is available for all protocols.
  - Persistent connections, so you can synchronize multiple times on the same connection.
  - Introduction of IPv6 support.
  - Improved error detection and debugging.

  For more information on using UltraLite as a client to MobiLink, see "UltraLite clients" [*UltraLite - Database Management and Reference*].

- **Set table order for synchronization** Synchronization from UltraLite clients now includes the ability to specify table ordering to avoid referential integrity issues during table upload. If you want to specify table order for synchronization, use the table_order synchronization parameter. See either "Additional Parameters synchronization parameter" [*UltraLite - Database Management and Reference*] or any of the following:
  - UltraLite.NET: "ULSyncParms class" [*UltraLite - .NET Programming*]
  - UltraLite C/C++: "ul_sync_info structure" [*UltraLite - C and C++ Programming*]
  - UltraLite for M-Business Anywhere: "SyncParms class" [*UltraLite - M-Business Anywhere Programming*]
  - UltraLite for embedded SQL: "ULGetSyncResult method" [*UltraLite - C and C++ Programming*]

Programming interfaces

General improvements

- **Cursor updates** UltraLite applications now support the ability to modify data in the database while processing a cursor. As with SQL Anywhere databases, not all query result sets allow cursor updates and deletes. Ensure you understand the cases in which cursor updates are allowed and executed. See "Fetching data" [*UltraLite - C and C++ Programming*].

- **Simplified connection strings** Because the default user ID of **DBA** and password of **sql** are always provided by UltraLite, you can now connect by specifying only the database in your connection string. Furthermore, most databases can be set with the DBF connection parameter. See "Connecting to an UltraLite database" [*UltraLite - Database Management and Reference*].

- **Introduction of MLFileTransfer functions** Use the file transfer function to download a file with the MobiLink file transfer utility. The file to be downloaded can be specific to a MobiLink username or a default file. For example, an application may choose to download a pre-configured empty

database file to replace the local database (at beginning of month or processing cycle). See "MobiLink File Transfer utility (mlfiletransfer)" [*MobiLink - Client Administration*].

- UltraLite for C/C++: "MLFileDownload method" [*UltraLite - C and C++ Programming*] and "MLFileUpload method" [*UltraLite - C and C++ Programming*]
- UltraLite.NET: "ULFileTransfer class" [*UltraLite - .NET Programming*] and "ULFileTransferProgressData class" [*UltraLite - .NET Programming*]
- UltraLite for M-Business Anywhere: Not applicable

- **Database creation** The UltraLite schema is now part of the database rather than in a separate *.usm* file. This means that applications can no longer create a new database in the same way as was supported in earlier versions.

  See any of the following:

  - UltraLite for C/C++: "ULCreateDatabase method" [*UltraLite - C and C++ Programming*]
  - UltraLite .NET: "ULDatabaseManager class" [*UltraLite - .NET Programming*]
  - UltraLite for M-Business Anywhere: "createDatabase method" [*UltraLite - M-Business Anywhere Programming*]

UltraLite C/C++

- **Support for Symbian OS** UltraLite C/C++ support is now provided for the Symbian OS platform, using CodeWarrior or Carbide C++ development environment.

- **New functions** Various new functions have been added in this release. These functions include:

  - The GetPublicationMask function gets the publication mask for a given publication name.

  - You must now call the appropriate ULEnable*Synchronization function before synchronizing over a specific network protocol. See "ULEnableHttpSynchronization method" [*UltraLite - C and C++ Programming*], "ULEnableAesDBEncryption method" [*UltraLite - C and C++ Programming*], "ULEnableAesFipsDBEncryption method" [*UltraLite - C and C++ Programming*], "ULEnableEccE2ee method" [*UltraLite - C and C++ Programming*], "ULEnableEccSyncEncryption method" [*UltraLite - C and C++ Programming*], "ULEnableRsaE2ee method" [*UltraLite - C and C++ Programming*], "ULEnableRsaFipsE2ee method" [*UltraLite - C and C++ Programming*], "ULEnableRsaFipsSyncEncryption method" [*UltraLite - C and C++ Programming*], "ULEnableRsaSyncEncryption method" [*UltraLite - C and C++ Programming*], "ULEnableTcpipSynchronization method" [*UltraLite - C and C++ Programming*], or "ULEnableZlibSyncCompression method" [*UltraLite - C and C++ Programming*].

- **Improved support for wide and narrow (ASCII) characters** Although UltraLite now has one database file format (narrow characters), applications can still use wide definitions of TCHAR. Wide characters are converted to their MBCS equivalent and vice versa as appropriate.

- **Enhanced functions changes** Enhancements to existing functions include:

- If your application does not require SQL support, using the ULInitDatabaseManagerNoSQL function instead of ULInitDatabaseManager can significantly reduce the size of the application.
- The SetReadPosition function has been enhanced to take a second parameter, offset_in_chars, which indicates if the offset is in bytes or characters.
- SetSynchInfo now performs an autocommit so all synchronization information is immediately saved.
- ULStoreDefragInit, ULStoreDefragFini and ULStoreDefragStep are no longer required. UltraLite now internally manages database store defragmentation.
- ULEnableUserAuthentication function is deprecated because user authentication is always enabled. UltraLite now permits the definition of up to four user names that may connect to the database (default is user name "DBA" with password "sql").
- MobiLink synchronization code must now invoke ULEnableTcpipSynchronization before invoking InitSynchInfo. See "InitSyncInfo method" [*UltraLite - C and C++ Programming*].

UltraLite embedded SQL

UltraLite embedded SQL is no longer a static API, and no longer requires a reference database. Instead, the SQL preprocessor requires only the source files. It generates functions that send SQL statements to UltraLite. Some SQL statements that were supported in previous releases are not supported by UltraLite SQL. Version 10 supports dynamic SQL statements which were not supported in previous releases.

- **New functions** Various new functions have been added in this release. These functions include:
  - The ULEnableZlibSyncCompression function enables zlib compression during synchronization. See "ULEnableZlibSyncCompression method" [*UltraLite - C and C++ Programming*] and "MobiLink client network protocol options" [*MobiLink - Client Administration*].

    Notes
    zlib compression is not supported for Palm OS or Symbian OS.

- **Enhanced functions** Enhancements to existing functions include:
  - GetSQLColumnName was added to UltraLite_RowSchema_iface. Depending on the type of schema, the function returns different results:
    - When used with a UltraLite_TableSchema, this function returns the column name specified by the column_id parameter.

      When used with a UltraLite_ResultSetSchema, this function returns either:
      - An alias name, if one is specified for the result set column in question.
      - The column name, if the result set column represents a column in a table.
      - An empty string in all other cases.

UltraLite.NET

UltraLite now supports ADO.NET 1.0 development in Visual Studio .NET 2003 and ADO.NET 2.0 development in Visual Studio 2005.

- **New methods** Various new functions have been added in this release. These functions include:
  - The ExecuteResultSet method executes a SQL SELECT statement and returns an updatable result set as a ULResultSet class. See "ExecuteResultSet method" [*UltraLite - .NET Programming*].
  - The ULResultSet class includes the following methods: Append*, Set*, Delete, Update. See "ULResultSet class" [*UltraLite - .NET Programming*] for details on these methods.
  - UltraLite.NET now supports TLS during TCP/IP synchronization.
  - ConnectionString properties and the ULConnectionParms object have been enhanced to support limited quoting. See "ULConnectionParms class" [*UltraLite - .NET Programming*].
  - The GetPublicationPredicate method returns publication predicate string for the specified publication. If the publication does not exist, SQLE_PUBLICATION_NOT_FOUND is set. See "GetPublicationPredicate method" [*UltraLite - .NET Programming*].
  - The SignalSyncIsComplete method signals the MobiLink provider for ActiveSync that an application has completed synchronization. See "SignalSyncIsComplete method" [*UltraLite - .NET Programming*].
  - The SetDatabaseOption method sets the value for the specified database option. See "SetDatabaseOption method" [*UltraLite - .NET Programming*].
- **Enhanced methods** Enhancements to existing methods include:
  - The ULSyncParms class now take a TableOrder order property to specify the order in which tables should be uploaded to the consolidated database. See "AdditionalParms property" [*UltraLite - .NET Programming*].
  - The GetSchemaTable method has now returns extended Table metadata. See "GetSchemaTable method" [*UltraLite - .NET Programming*] for a complete list.
  - The UpdateBegin method is now an optional at the ResultSet level when a table is in UL_TABLE_ACCESS_MODE_NONE or UL_TABLE_ACCESS_MODE_FIND_AGAIN. This change was required to make the UltraLite.NET API compatible with the ADO.NET 2.0 result set. See "UpdateBegin method" [*UltraLite - .NET Programming*].
  - The GetDatabaseProperty method now recognizes more properties. See "GetDatabaseProperty method" [*UltraLite - .NET Programming*].
  - The ULSyncProgressData class now includes a Flags property. See "Flags property" [*UltraLite - .NET Programming*].

UltraLite for AppForge Crossfire

UltraLite for AppForge now supports the Symbian OS platform. Support for the UltraLite engine has been added in this release, allowing multiple applications to concurrently access a single database.

- **New method** The OnWaiting method provides a mechanism for the user application to process GUI events and possibly cancel the current operation.

UltraLite for M-Business Anywhere

- **New methods** Various new methods have been added in this release. These functions include:
  - The setMBAServerWithMoreParms method sets proxy server information when using one-button synchronization. This new method enhances the existing setMBAServer method, by adding a new string argument named **additional**.
  - The getPublicationMask method gets the publication mask for a given publication name.
  - The getPublicationPredicate method returns publication predicate string for the specified publication. If the publication does not exist, SQLE_PUBLICATION_NOT_FOUND is set.
- **Enhanced methods** Enhancements to the following existing method includes:
  - The setStream method now supports ECC (Elliptic curve cryptography) for TLS (Transport Layer security). See "SyncParms class" [*UltraLite - M-Business Anywhere Programming*].

Note
ECC encryption is not available on all platforms. For a list of supported platforms, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

Behavior changes and deprecated features

Following is a list of changes to UltraLite introduced in version 10.0.0.

Deprecated platforms

- Support for the PocketPC 2000 OS has been deprecated for this release.
- Support for CodeWarrior 8 has been removed. You must use Code Warrior 9 instead.
- Support for Windows CE MIPS processors have been removed.

Removed components, modules, namespaces

The following programming interfaces have been dropped from this release:

- **UltraLite for ActiveX** All applications must be rewritten using a supported API.

- **Static Java API** All applications must be rewritten using a supported API.
- **Native UltraLite for Java** All applications must be rewritten using a supported API.
- **Static C++ API and Static embedded SQL** Developers wanting to write C++ applications must program using the dynamic C++ interface. If you have an application written with the static C++ library from previous versions, UltraLite 10 includes a migration utility to simplify the move to this new library. See "Upgrading UltraLite" on page 329.
- **iAnywhere.UltraLite namespace** In UltraLite.NET, this namespace is no longer supported. You must re-write your applications using the iAnywhere.Data.UltraLite namespace instead.

Removed utilities

- **Schema Painter** Because you no longer need a schema file to create an UltraLite database, the Schema Painter tool has been removed.
- **Database conversion tool** The Database conversion tool (the ulconv utility) is no longer supported. For the ulconv functionality, use the ulcreate, ulload, ulsync, and ulunload utilities.
- **ulxml utility** The ulxml utility that converted schema files to XML is no longer supported. For similar ulxml functionality, use ulload and ulunload to convert databases to XML instead.
- **ulisql** The ulisql utility is no longer supported. Instead, Interactive SQL (dbisql) now supports UltraLite.
- **ulgen** The ulgen utility is no longer supported. For UltraLite deployments that used this utility, you need to upgrade your database and C/C++ applications. See "Upgrading UltraLite" on page 329.

Removed, deprecated, and modified functions

- **UltraLite for C/C++ API** Changes to functions and macros in the C/C++ API include:
  - The database schema can no longer be connected to nor upgraded dynamically because the *.usm* file no longer exists. All classes and functions relating to this former feature of UltraLite have been removed.
  - ULEnablePalmRecordDB and ULEnableFileDB have been removed in this version.
  - All ULEnableXXXX functions must now be called with an initialized SQLCA.
  - The macro UL_STORE_PARMS has been deprecated in release 10. Connection and creation options are specified in the appropriate parameter when calling OpenConnection or CreateDatabase.
  - ULSecureCerticomTLSStream and ULSecureRSATLSStream are deprecated in this release. In their place, you can use ULEccTlsStream and ULRsaTlsStream.
  - The security and security_parms fields of ul_sync_info are removed. Instead, set the stream field to the appropriate string value: tcpip, http, https or tls. Additionally, combine the security parameters with the other stream parameters. TCPIP is always the underlying transport mechanism and TLS over HTTP is no longer supported. Instead you can use the HTTPS synchronization

stream. See "UltraLite synchronization parameters and network protocol options" [*UltraLite - Database Management and Reference*].

- ULSocketStream, ULHTTPStream and ULHTTPSStream have been changed to return the appropriate string value that is now required.
- ULActiveSyncStream is removed for Windows CE devices. An UltraLite application that has registered with the ActiveSync provider must instead use one of the standard synchronization streams when it receives the synchronize message in its Windows message handler.

- **Embedded SQL** Changes to functions in the embedded SQL interface to the C/C++ API include:
  - The database schema can no longer be upgraded dynamically because the *.usm* file no longer exists. All classes and functions relating to this former feature of UltraLite have been removed.
- **UltraLite.NET API** Changes to functions in the UltraLite.NET API include:
  - The database schema can no longer be connected to nor upgraded dynamically because the *.usm* file no longer exists. All classes and methods relating to this former feature of UltraLite have been removed.
  - ParmsUsed property has been renamed ToString in the ULConnectionParms class.
  - GetSQLColumnName has been renamed to GetColumnSQLName.
  - ULStreamType members UNKNOWN and ACTIVE_SYNC are removed from this enumeration. The default is now ULStreamType.TCPIP.
- **UltraLite for MobileVB API** Changes methods in the MobileVB API include:
  - The database schema can no longer be connected to nor upgraded dynamically because the *.usm* file no longer exists. All classes and methods relating to this former feature of UltraLite have been removed.
- **UltraLite for M-Business Anywhere API** Changes to functions in the M-Business Anywhere API include:
  - The database schema can no longer be connected to nor upgraded dynamically because the *.usm* file no longer exists. All classes and methods relating to this former feature of UltraLite have been removed.

Name changes

- **ULUtil** The ULUtil utility for Palm OS has been renamed ULDBUtil.
- **ulmvbreg** ulmvbreg has been renamed ulafreg. This utility is now installed to the *install-dir\win32* directory.

Miscellaneous

- **ulcond.log** Version 10 of the UltraLite HotSync conduit installer (ulcond10) no longer writes messages to this log file.

Sybase Central and Interactive SQL

The following sections describe the new features, behavior changes, and deprecated features in Sybase Central and Interactive SQL for version 10.0.0.

New features

Following is a list of additions to Sybase Central and Interactive SQL introduced in version 10.0.0.

Sybase Central

This section describes new features in Sybase Central. Changes and additions made to the Sybase Central plug-ins are described in the following sections:

- "SQL Anywhere plug-in" on page 293
- "New plug-ins for Sybase Central" on page 294

- **Sybase Central Task list** You can choose to view tasks in the left pane of Sybase Central, instead of a tree structure of the database. The Task list shows common tasks related to the object that is currently selected. The Task list includes common tasks, navigation options, and links to the documentation. See "Navigating Sybase Central" [*SQL Anywhere Server - Database Administration*].

- **New Connections menu** In previous releases, you could press F11 to open the New Connection window that let you choose the plug-in you wanted to connect with. Sybase Central now provides a Connections menu where you can choose the plug-in you want to use. The New Connection window has been removed, but pressing opens the Connections menu where you can choose the plug-in you want to use for your connection.

- **Connection profile enhancements** You can now add descriptions to Sybase Central connection profiles. Connection profiles can also be imported and exported.

- **Plug-in searches** You can now search for objects that contain specified text within the plug-ins and databases in Sybase Central by choosing View » Search Pane.

- **Context dropdown list** The new Context dropdown list shows you the object that is currently selected in the object tree to make it easier to navigate through plug-ins, especially when the object tree is not open in the left pane.

SQL Anywhere plug-in

- **Deadlocks database tab** When you are connected to a SQL Anywhere database in Sybase Central, you can view information about deadlocks on the Deadlocks tab. See "Viewing deadlocks from Sybase Central" [*SQL Anywhere Server - SQL Usage*].

- **Entity-relationship diagrams** Sybase Central displays entity-relationship diagrams for databases, showing the tables in the database and their foreign key relationships. See "Viewing entity-relationship diagrams from the SQL Anywhere 12 plug-in" [*SQL Anywhere Server - Database Administration*].

- **New Create Schedule wizard and other enhancements to the Events folder** For each scheduled event, the Events folder now displays the next scheduled time when the event will be triggered. For each conditional event, the folder displays the system event and optionally the trigger conditions that trigger the event. Schedules are created using the new Schedule Creation wizard. When creating a new scheduled event, the Event wizard allows you to create a single schedule, but you can later add additional schedules to an event if required. See "Defining schedules" [*SQL Anywhere Server - Database Administration*].

- **Maintenance plans** You can set up a schedule for validating and backing up a database automatically and have the output log emailed to you. See "Creating a maintenance plan" [*SQL Anywhere Server - Database Administration*].

- **New Restore Database wizard** You can now use the Restore Database wizard to restore a database from an archive backup. See "Restore from an archive backup" [*SQL Anywhere Server - Database Administration*].

- **Editor enhancements for Sybase Central and Interactive SQL** You can choose the font used in the editor windows of Sybase Central and Interactive SQL on the Format tab of the Options window.

  A typing completion option has been added to the editor for database object names. See "Using text completion" [*SQL Anywhere Server - Database Administration*].

New plug-ins for Sybase Central

- **QAnywhere plug-in** The QAnywhere plug-in provides an easy-to-use graphical interface for creating and administering your QAnywhere applications.

  See "New QAnywhere plug-in for Sybase Central" on page 273.

- **UltraLite plug-in** The UltraLite plug-in allows you to create, modify, and administer your UltraLite databases in a graphical user interface.

  See "Graphical administration tools" on page 282.

- **MobiLink Create Synchronization Model wizard and Model mode** You can now create a synchronization model using a wizard, and edit your model in the MobiLink plug-in using the new Model mode. You can also set up MobiLink server-initiated synchronization. The old features of the MobiLink plug-in have also been enhanced and are preserved in Admin mode.

  See "Enhancements to the MobiLink plug-in for Sybase Central" on page 251.

Interactive SQL

- **Interactive SQL can connect to UltraLite databases** You can now use Interactive SQL to develop and test SQL statements with UltraLite databases. The ulisql utility has been deprecated. See "Graphical administration tools" on page 282.

- **Interactive SQL integrates with third party source control systems** Interactive SQL can integrate with third party source control systems, allowing you to perform a number of common source control operations on files from within Interactive SQL, such as checking in, checking out, and comparing against old versions. See "Using source control integration" [*SQL Anywhere Server - Database Administration*].

- **New Interactive SQL options** The isql_maximum_displayed_rows option lets you specify the number of rows that appear in the result set in Interactive SQL, while the isql_show_multiple_result_sets option specifies whether multiple result sets can appear in the Results pane in Interactive SQL. See "isql_maximum_displayed_rows option [Interactive SQL]" [*SQL Anywhere Server - Database Administration*] and "isql_show_multiple_result_sets [Interactive SQL]" [*SQL Anywhere Server - Database Administration*].

- **Text completion** Interactive SQL now includes a typing completion option that can fill in the names of the following object types: tables, views, columns, stored procedures, and system functions. See "Using text completion" [*SQL Anywhere Server - Database Administration*].

- **DESCRIBE statement now supported by Interactive SQL** The DESCRIBE statement enables you to obtain the following information about a specified table or procedure:

  - all columns found in the table
  - all indexes found in the table
  - all parameters used with the stored procedure

  See "DESCRIBE statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

- **Interactive SQL supports the @data option** When starting Interactive SQL from a command prompt, you can specify the **@***data* option to read in options from the specified environment variable or configuration file. See "Interactive SQL utility (dbisql)" [*SQL Anywhere Server - Database Administration*].

SQL Anywhere Console utility

- **SQL Anywhere Console supports the @data option** When starting the SQL Anywhere Console from a command prompt, you can specify the **@***data* option to read in options from the specified environment variable or configuration file. See "SQL Anywhere Console utility (dbconsole)" [*SQL Anywhere Server - Database Administration*].

Behavior changes and deprecated features

Following is a list of changes to Sybase Central and Interactive SQL introduced in version 10.0.0.

- **Interactive SQL no longer sets the quoted_identifier option to On** In previous versions of the software, Interactive SQL set the quoted_identifier option to On. It now uses the database's setting

for this option (by default, this option is set to On). See "quoted_identifier option" [*SQL Anywhere Server - Database Administration*].

- **isql_plan option no longer supports the NONE parameter** The NONE parameter is no longer supported by the isql_plan option.

- **Interactive SQL can return unlimited result sets** In previous versions of the software, if you executed a query that returned multiple result sets, Interactive SQL only displayed a maximum of 10 result sets. Now Interactive SQL displays all result sets returned by the query. See "isql_show_multiple_result_sets [Interactive SQL]" [*SQL Anywhere Server - Database Administration*].

- **Interactive -f option behavior change** When starting Interactive SQL with the -f option, a connection is not made to a database automatically. Previously, a connection was opened automatically.

- **Accessing and saving graphical plans in Interactive SQL**

  - Two new menu choices, Open Plan and Save Plan, are available from the File menu in Interactive SQL for opening and saving graphical plans (previously this was done using the same Open and Save menu items as you used for opening and saving the SQL statements.

  - Previously, graphical plans were saved with an *.xml* file extension. Now, they are saved with the extension *.saplan*. However, the previous *.xml* file extension is still supported for displaying graphical plans that were stored with that extension.

  - In the Options window, you can now make Interactive SQL the default editor for both *.sql* and *.saplan* (graphical plan) files.

- **SET OPTION statement PUBLIC keyword deprecated** The PUBLIC keyword is deprecated for setting Interactive SQL options using the SET OPTION statement. See "Interactive SQL options" [*SQL Anywhere Server - Database Administration*].

- **EXIT statement now closes the current Interactive SQL window** In previous releases, executing an EXIT statement from Interactive SQL closed all the Interactive SQL windows. Now, only window where the statement is executed is closed. See "EXIT statement [Interactive SQL]" [*SQL Anywhere Server - SQL Reference*].

- **New error code for SQLE_ENGINE_NOT_MULTIUSER** Applications that were programmed to handle SQLE_ENGINE_NOT_MULTIUSER now need to check for a new error code. Previously, if an application attempted a write operation on the database while another thread was sending an upload to MobiLink, the runtime would return SQLE_ENGINE_NOT_MULTIUSER. Now the runtime will return a new, more accurate error code: SQLE_ULTRALITE_WRITE_ACCESS_DENIED. See "Write access was denied" [*Error Messages*].

- **Sybase Central plug-in Utilities tabs removed** The Utilities tabs in the SQL Anywhere, MobiLink, and UltraLite plug-ins have been replaced with a Tools button. You can also access utilities from the Sybase Central Tools menu.

Deprecated and discontinued features

- **jConnect no longer supported for connecting to Sybase Central, Interactive SQL, or the SQL Anywhere Console** Sybase Central, Interactive SQL, and the SQL Anywhere Console utility (dbconsole) no longer support connections to SQL Anywhere databases using jConnect. You can still use the iAnywhere JDBC driver to connect to databases from these applications. As a result of this change, the following features have been removed:
  - The -jconnect and -odbc options for the Interactive SQL utility (dbisql) have been removed.
  - The -jconnect and -odbc options for the SQL Anywhere Console utility (dbconsole) have been removed.
  - The Connect window used for connecting to Interactive SQL, the SQL Anywhere Console, and the SQL Anywhere and MobiLink plug-ins in Sybase Central no longer allows you to specify whether jConnect or the iAnywhere JDBC driver is used. The iAnywhere JDBC driver is used for all connections.
- **UltraLite plan now appears on the Plan tab in Interactive SQL** In previous releases, the Interactive SQL Results pane had an UltraLite Plan tab that displayed the UltraLite plan optimization strategy. The UltraLite Plan tab has been removed, so now when you are connected to an UltraLite database from Interactive SQL, the plan appears on the Plan tab.
- **Sybase Central SQL Anywhere plug-in no longer supports version 7 databases** Support for version 7 database servers and databases created with version 7 software has been removed from the SQL Anywhere plug-in. When unloading and reloading the database into a reload file, or into a new or existing database, you can still connect to a database created with version 5, 6, or 7 software running on a version 8 or later database server. See "Rebuilding version 9 and earlier databases for version 12" on page 309.
- **isql_log option deprecated** The isql_log option for logging statements executed during an Interactive SQL session is deprecated. Use the START LOGGING and STOP LOGGING statements instead. See "Logging commands" [*SQL Anywhere Server - Database Administration*].
- **Sybase Central file name changes** In addition to file changes mentioned for Sybase Central, the following files have been added:

New name	Old name
asaplugin.jar	*saplugin.jar*

  The registry key has also changed to reflect the new version of Sybase Central, and is now:

```
HKLM\SOFTWARE\Sybase\Sybase Central\5.0 (registry entries)
```

Documentation enhancements

The documentation for pre-existing features has been enhanced in several areas, including the following:

- **Context-sensitive help for SQL statements** In Interactive SQL, you can now right-click a SQL statement name to open the reference topic for the statement.

- **Getting Started with MobiLink** A new introductory book has been added that helps new users understand how to develop distributed applications with MobiLink.

  See "MobiLink - Getting Started".

- **New chapter about SQL Anywhere administration tools in Database Administration Guide** A new chapter has been added that focuses on how to use Sybase Central, Interactive SQL, and the SQL Anywhere Console utility.

  See "SQL Anywhere graphical administration tools" [*SQL Anywhere Server - Database Administration*].

- **Supported platform information updated for EBFs** There are now supported platform pages that are installed with the product and that refer to the build you install. These pages are installed to the *documentation\en* subdirectory of your SQL Anywhere installation. The pages are updated in EBFs. The documentation links to these pages where required.

- **SQL Anywhere security features moved** The book *SQL Anywhere Studio Security Guide* has been removed. SQL Anywhere security features are now documented in "SQL Anywhere Server - Database Administration".

- **ODBC Drivers book removed** The book *ODBC Drivers for MobiLink* has been removed.

Product-wide features

The following sections describe the new features, behavior changes, and deprecated features that affect all components of SQL Anywhere version 10.0.0.

New features

Following is a list of product-wide additions introduced in version 10.0.0.

- **Product name changed to SQL Anywhere 10** SQL Anywhere Studio has been renamed SQL Anywhere 10, and Adaptive Server Anywhere has been renamed SQL Anywhere. Many names of files, directories, services, and executables have consequently changed, mostly to reflect the change from ASA to SA. These changes are detailed in the relevant Behavior Change topics of this chapter.

- **New install for DataWindow .NET** DataWindow .NET is a custom control tool that is useful to database developers using Visual Studio. It is provided as an optional component during your SQL Anywhere installation. Complete documentation is provided in the installation.

- **RSA now included with SQL Anywhere** You no longer have to purchase a separate license to use RSA encryption. See "Transport-layer security" [*SQL Anywhere Server - Database Administration*].

- **Feature statistics collection** SQL Anywhere 10 tracks information about the computer running the software, such as the operating system, the options use to start the database server, the SQL Anywhere 10 software build being used, and so on, as well as information about the SQL Anywhere 10 features being used. If a fatal error occurs, the software automatically prompts you to send any information about the problem, as well as the SQL Anywhere feature statistics to iAnywhere. This information can be used to help diagnose the problem, should you open a technical support case. See "Error reporting in SQL Anywhere" [*SQL Anywhere Server - Database Administration*].

  You also have the option of sending the feature statistics at any time using the Support utility (dbsupport). The feature statistics information helps iAnywhere understand how the product is being used to help improve the software. See "Support utility (dbsupport)" [*SQL Anywhere Server - Database Administration*].

  The SADIAGDIR environment variable specifies the location of the directory where crash reports and feature statistics are stored. See "SADIAGDIR environment variable" [*SQL Anywhere Server - Database Administration*].

- **Error reporting** When an internal error occurs in Sybase Central, Interactive SQL, or the SQL Anywhere Console utility, a log of the error is written to the hard drive and a window appears where you can choose to send the error report to iAnywhere.

  In addition, if a fatal error occurs in any of the following: personal server, network server, MobiLink server, dbmlsync, MobiLink Listener, QAnywhere agent, SQL Remote, or Replication Agent, a log of the error is written to the hard drive and a window appears where you can choose to send the error report to iAnywhere. The Support utility (dbsupport) can be configured to automatically submit these error reports. See "Support utility (dbsupport)" [*SQL Anywhere Server - Database Administration*].

  If you choose not to submit an error report, the file remains in the diagnostic directory on your hard disk. The SADIAGDIR environment variable specifies the location of the directory where crash reports and feature statistics are stored. See "SADIAGDIR environment variable" [*SQL Anywhere Server - Database Administration*].

- **Control which features are available in the administration tools** You can now control which features are available for users in the administration tools using an *OEM.ini* file located in the same directory as the tool's *.jar* file.

- **Using SQL Anywhere 10 on Windows Vista** The following known issues exist on Windows Vista:

  - Updating license information with the Server Licensing utility (*dblic.exe*) fails unless you have administrator privileges or write permissions on the directory containing the server executables.

  - There is a known problem with using shared memory when running SQL Anywhere as a service or an unprivileged account. This problem is under investigation. As an alternative, you can use TCP/IP.

Behavior changes

Following is a list of product-wide changes in version 10.0.0.

- **Location of samples directory changed** The samples included with SQL Anywhere 10 are no longer installed under your SQL Anywhere 10 installation directory. As a result of this change, the sample databases are now installed to the following locations:

Sample database	Location
SQL Anywhere sample database	*samples-dir\demo.db*
MobiLink CustDB sample consolidated database application	*samples-dir\MobiLink\CustDB*
UltraLite CustDB sample database	*samples-dir\UltraLite\CustDB*

  For a list of the default location of *samples-dir* for each supported operating system, see "Samples directory" [*SQL Anywhere Server - Database Administration*].

- **Unsupported platforms** The following platforms are no longer supported:

  - Windows 95
  - Windows 98
  - Windows Me
  - Windows NT
  - Compaq Tru64

  For other changes to platform support, see "Supported platforms" [*SQL Anywhere 12 - Introduction*].

- **Sample database revised and renamed** The SQL Anywhere sample database is now called *demo.db*. Objects such as tables, columns, views, and indexes now have full-word names to benefit users who use screen readers. See "SQL Anywhere sample database" [*SQL Anywhere 12 - Introduction*].

- **Miscellaneous file name changes** In addition to file name changes mentioned for individual products, the following file names have changed:

Old name	New name
asa.cvf	*sqlany.cvf*
asaldap.ini	*saldap.ini*
asasrv.ini	*sasrv.ini*
asa_config.sh	*sa_config.sh*
install-dir/SYBSasa9/lib	*sqlanywhere10/lib32* or *sqlanywhere10/lib64*
install-dir/SYBSasa9/bin	*sqlanywhere10/bin32* or *sqlanywhere10/bin64*

To reflect the new product name, some registry keys have changed. They are now:

```
HKLM\SYSTEM\CurrentControlSet\Services\Eventlog\Application\SQLANY
HKLM\SYSTEM\CurrentControlSet\Services\Eventlog\Application\SQLANY 10.0
```



```
HKLM\SYSTEM\CurrentControlSet\Services\Eventlog\Application\SQLANY 10.0
Admin
HKLM\SOFTWARE\Sybase\Sybase Central\5.0 (registry entries)
HKLM\SOFTWARE\Sybase\SQL Anywhere\10.0 (registry entries)
```

In addition to file changes mentioned for individual products, the following file has been added: *ulbase.lib*.

The following service group names have changed:

Description	Old name	New name
Network server	ASANYServer	SQLANYServer
Personal server	ASANYEngine	SQLANYEngine
MobiLink synchronization client	ASANYMLSync	SQLANYMLSync
Replication Agent	ASANYLTM	SQLANYLTM

Changes to ODBC drivers used by MobiLink, QAnywhere, and remote data access

- **Sybase Adaptive Server Enterprise driver** SQL Anywhere no longer includes an iAnywhere Solutions ODBC driver for Adaptive Server Enterprise. Instead, the Adaptive Server Enterprise native driver is tested to work with MobiLink. The iAnywhere Solutions 9 - Adaptive Server Enterprise Wire Protocol driver is no longer supported.

  See http://www.ianywhere.com/developer/technotes/odbc_mobilink.html.

- **IBM UDB DB2 driver** SQL Anywhere no longer includes an iAnywhere Solutions ODBC driver for DB2. Instead, the IBM DB2 8.2 CLI driver is tested to work with MobiLink. This native DB2 driver supports DB2 versions 8.1 and 8.2. The following drivers are no longer supported: IBM DB2 7.2 ODBC driver and iAnywhere Solutions 9 - DB2 Wire Protocol driver.

  See http://www.ianywhere.com/developer/technotes/odbc_mobilink.html.

- **Oracle driver** The iAnywhere Solutions 10 - Oracle Wire Protocol driver is available by separate download.

  See http://www.ianywhere.com/developer/technotes/odbc_mobilink.html.

Upgrading to SQL Anywhere 12

Features requiring an unload/reload, upgraded database, or updated client libraries

Many features are available when you upgrade your SQL Anywhere database server. While many features are available when you run an older database on the latest version of the database server, to access some features, you must unload and reload your database, perform an upgrade on the database file, or update your client libraries.

Before version 10, the SQL Anywhere database server was called Adaptive Server Anywhere.

Note
Features not listed in the categories below only require a SQL Anywhere 12.0.0 database server, they do not require the database to be upgraded or the client libraries to be updated. For a complete list of new features in SQL Anywhere 12, see "What's new in version 12.0.0" on page 1.

Features that require an unload/reload of your database

- General performance enhancements. See "Performance enhancements" on page 26.
- **UltraLite**: UltraLite version 12 cannot connect to databases created with earlier versions of UltraLite. Therefore, an unload/reload is required. However, UltraLite version 11 can synchronize with MobiLink version 12.

Features that require only an upgrade (dbupgrad utility or UPGRADE DATABASE statement)

Features that require only an upgraded database also work if you unload/reload your database.

- Indexing performance enhancements. See "Improved index performance in SQL Anywhere 12.0.0" on page 27.
- Spatial data support features. See "Support for spatial data" on page 3.
- TIMESTAMP WITH TIME ZONE. See "New TIMESTAMP WITH TIME ZONE data type" on page 22.

Features that require updated client libraries

- Connection pooling. See "SQL Anywhere connection pooling" [*SQL Anywhere Server - Database Administration*].
- Progress messages. See "New progress_messages option" on page 13.
- Idle timeout on shared memory connections. See "Behavior change for the Idle connection parameter" on page 31.

- TIMESTAMP WITH TIME ZONE. See "New TIMESTAMP WITH TIME ZONE data type" on page 22.
- Spatial data support on the client side. See "Support for spatial data" on page 3.
- SQL Anywhere JDBC driver. See "New SQL Anywhere TYPE-2 JDBC driver" on page 23.

Upgrading SQL Anywhere

Before using existing applications with this version of the software, be sure to review the list of behavior changes to determine whether your application is affected. See "SQL Anywhere 12 - Changes and Upgrading" on page 1.

Upgrading version 10 and later databases

If you are upgrading from version 10 or later, you can either upgrade or rebuild your database. Upgrading or rebuilding is an optional step because the version 12 software can be used with a version 10 or later database. However, if you want to take advantage of all the new features in version 12, you must rebuild your database. See "Upgrading version 10 and later databases" on page 315 and "Rebuilding version 10 and later databases" on page 307.

Databases with materialized views
It is recommended that you refresh the materialized views in your database after upgrading your database server, or after rebuilding or upgrading your database to work with an upgraded database server. See "Refresh manual views" [*SQL Anywhere Server - SQL Usage*].

Upgrading version 9 and earlier databases

If you are upgrading to version 12 from version 9 or earlier, you must rebuild the database, which consists of unloading the old database, and reloading it into a new version 12 database. Attempting to start version 9 or earlier databases results in an error on database startup. There are several approaches for rebuilding existing databases:

- Use the version 12 Unload utility (dbunload) with the -an (create a new database) or -ar (replace the old database) option. See "Rebuild a version 9 or earlier database using the Unload utility" on page 312.

  Note
  The Unload utility (dbunload) has the same file name in all versions of SQL Anywhere. You must make sure you are using the correct version. Run the command `dbunload -?` to determine which version of the Unload utility you are using. See "Using the utilities" on page 305.

- Unload the database using the version 12 Unload utility, and then reload the database using the *reload.sql* file on the version 12 database server.

  If you need to make schema changes, this is the recommended way of upgrading. After you make the schema changes, you can create a new database, and then apply the reload script to it.

- Use the **Unload Database Wizard** in Sybase Central. You can choose to create a new database, replace an existing database with the new database, or unload the database to a file. See "Rebuild a version 9 or earlier database from Sybase Central" on page 311.

- Unload the database using an older version of dbunload, and then reload the database using the *reload.sql* file and the version 12 database server. You should only use this approach if the other methods fail because deprecated or unsupported database option settings, objects, or SQL syntax could be unloaded into the *reload.sql* file. You must edit the file manually if problems occur during the reload. The internal reload capabilities of version 12 take care of many of these problems.

Rebuilding Mac OS X databases
SQL Anywhere 9.0.2 for Mac OS X was supported on PPC, while SQL Anywhere 10 and later for Mac OS X are supported on Intel. If you have a version 9.0.2 or earlier database on Mac OS X, you have two options for unloading the database:

- Unload the database using the version 9.0.2 software.

- Copy the database to a different platform where SQL Anywhere 12 is installed, and then unload the database using the version 12 software.

Once the database is unloaded, you can perform the reload on Mac OS X using the version 12 software.

If you want to change the characteristics of the database during unload and reload (for example, change a case-sensitive database to a case-insensitive database), the procedure is more involved. See "Rebuilding databases" [*SQL Anywhere Server - SQL Usage*].

Compatibility with existing software

- SQL Anywhere 12 database servers support connections from client applications using software from version 6.0.0 or later. Version 5 and earlier clients cannot connect to a version 12 database server.

- You can manage old databases and old database servers from the current version of Sybase Central as follows:

  - You can connect to and administer version 10 and later databases running on version 10 and later database servers.

  - You can connect to a version 5 or later database running on the same computer as Sybase Central to rebuild the database using the **Unload Database Wizard** from Sybase Central. The database is stopped before it is unloaded.

  - There is no support for version 9 and earlier databases running on version 9 and older database servers.

Using the utilities

If you have multiple versions of SQL Anywhere on your Windows computer, you must pay attention to your system path when using utilities. Since the installation adds the most recently installed version

executable directory to the end of your system path, it is possible to install a new version of the software, and still inadvertently be running the previously installed version.

For example, if a SQL Anywhere version 8 executable directory is ahead of the SQL Anywhere 12 executable directory in your path and you use the dbinit command, you will use the version 8 utility, and consequently create a version 8 database.

There are several ways you can ensure that you are using the version 12 utilities, including:

- Modify your system path so that the SQL Anywhere 12 executable directory is before any previous version executable directory.
- Change to the SQL Anywhere 12 executable directory before executing your command.
- Specify a fully-qualified path name to the utility name that indicates the exact location of the utility you want to run.
- Create scripts to change your environment to use the correct version of the utilities.
- Uninstall the old software.

Important upgrade precautions

There are several precautions you should take before upgrading SQL Anywhere:

- **Check the behavior changes** Confirm that none of the documented behavior changes affect your application. If they do, you must update your application. See "What's new in version 12.0.0" on page 1.
- **Test your application** Test your application thoroughly in a SQL Anywhere 12 environment before upgrading any applications in production use.
- **Use the correct version of the utilities** Make sure you use the correct version of the database utilities with your new database. See "Using the utilities" on page 305.
- **Validate and back up the database** Before you begin an upgrade, validate your database, and back up your software and database. To ensure future recoverability, back up the database when you finish the upgrade.
- **Synchronize before upgrading** For databases involved in synchronization, such as UltraLite databases or SQL Anywhere remote databases in MobiLink installations, you must perform a successful synchronization before upgrading.
- **Test your upgrade procedure** Test your upgrade procedure carefully before carrying it out on a production system.

SQL Anywhere is used in many different configurations, and no upgrade guidelines can be guaranteed for all cases.

Upgrade quick start

For previous users of the software, the following steps summarize the process for rebuilding your database to version 12.

To rebuild a database (command line)

1. Back up the database. For example:

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```

   For more information, see "Backup quick start" [*SQL Anywhere Server - Database Administration*].

2. If possible, defragment the drive where the new database will be stored because a fragmented drive can decrease database performance.

3. Shut down all SQL Anywhere database servers because the version 12 dbunload utility cannot be used against a database that is running on a previous version of the database server. For example:

   ```
   dbstop -c "DBF=mydb.db;UID=DBA;PWD=sql"
   ```

4. Unload and reload (rebuild) the old database into a new version 12 database. For example:

   ```
   dbunload -c "DBF=mydb.db;UID=DBA;PWD=sql" -an mydb12.db
   ```

5. Back up the new database before using it. For example:

   ```
   dbbackup -c "DBF=mydb12.db;UID=DBA;PWD=sql" new-db-backup-dir
   ```

6. Validate the new database before using it. For example:

   ```
   dbvalid -c "DBF=mydb12.db;UID=DBA;PWD=sql"
   ```

See also

- "Rebuild a version 9 or earlier database using the Unload utility" on page 312
- "Rebuild a version 9 or earlier database from Sybase Central" on page 311

Rebuilding version 10 and later databases

Rebuilding a database consists of unloading and reloading the database to upgrade its file format. When you upgrade the file format, it changes the format used to store and access data on disk, letting you use all the new features and performance enhancements in the latest version of the software.

Caution
Unloading and reloading a large database can be time consuming and can require a large amount of disk space. The process may require disk space approximately twice the size of your database to hold the unloaded data and the new database file.

If you are rebuilding a database that is a remote database in a MobiLink installation or that is involved in SQL Remote replication, and if you use the dbunload utility, you must be sure to use the -ar or -an option.

These options ensure that the transaction log offsets for the new database are set to match those of the old database.

When using dbunload with a version 10 or later database, the version of dbunload used must match the version of the database server used to access the database. If an older version of dbunload is used with a newer database server, or vice versa, an error is returned.

Because of index changes in SQL Anywhere, when you rebuild a database by unloading and reloading it, the rebuilt database may be smaller than the original database. This decrease in database size does not indicate a problem or a loss of data.

Note
It is recommended that you back up your database before you rebuild it.

Reloading tables with autoincrement columns

You can retain the next available value for autoincrement columns in the rebuilt database by specifying the dbunload -l option. This option adds calls to the sa_reset_identity system procedure to the generated *reload.sql* script for each table that contains an autoincrement value, preserving the current value of SYSTABCOL.max_identity.

Rebuild the database

To rebuild a database (Sybase Central)

1. Carry out the standard precautions for upgrading software. See "Important upgrade precautions" on page 306.
2. Back up the database. For example:

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```

   For more information, see "Backup quick start" [*SQL Anywhere Server - Database Administration*].
3. Choose **Start** » **Programs** » **SQL Anywhere 12** » **Administration Tools** » **Sybase Central**.
4. Start a version 12 database server running the database you want to upgrade, and then connect to the database from Sybase Central.
5. Choose **Tools** » **SQL Anywhere 12** » **Unload Database**.
6. Read the text on the first page of the **Unload Database Wizard** and then click **Next**.
7. Select **Unload A Database Running On A Current Version Of The Server**, and then select the database from the list. Click **Next**.
8. Choose to unload and reload into a new database. Click **Next**.
9. Specify a new file name for the database.
10. You can also specify the page size for the new database, but the page size you specify cannot be larger than the database server page size. The default page size is 4096 bytes. You can encrypt the database

file if you want. If you choose strong encryption, you need the encryption key each time you want to start the database. Click **Next**.

For more information about database file encryption, see "Encrypting and decrypting a database" [*SQL Anywhere Server - Database Administration*].

11. Choose **Unload Structure And Data**. You can also select any other options you want for your database. Click **Next**.
12. Choose **Unload All Database Objects**. Click **Next**.
13. Specify whether you want to connect to the new database when the unload/reload is complete.
14. Click **Finish** to start the process. You should examine the new database to confirm that the rebuild completed properly.

    For more information about using the **Unload Database Wizard**, see "Export data with the Unload Database Wizard" [*SQL Anywhere Server - SQL Usage*].

To rebuild a database (command line)

1. Carry out the standard precautions for upgrading software. See "Important upgrade precautions" on page 306.
2. Ensure that you have exclusive access to the database to be upgraded and ensure that the path of the version 12 utilities is ahead of the path of the other utilities in your system path. See "Using the utilities" on page 305.
3. Run the Unload utility (dbunload) and use the -an option to create a new database.

   ```
   dbunload -c "connection-string" -an new-db-file
   ```

   The database user specified in the *connection-string* must have DBA authority on the database that is being rebuilt.

   This command creates a new database. If you want to replace the existing database with an upgraded database, use the -ar option instead of -an. To use the -ar option, you must connect to a personal database server, or to a network database server on the same computer as the Unload utility (dbunload).

   For information about other Unload utility (dbunload) options, see "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].
4. Shut down the database and archive the transaction log before using the reloaded database.

   If you want to change the characteristics of the database during unload and reload (for example, change a case-sensitive database to a case-insensitive database), the procedure is more involved. See "Rebuilding databases" [*SQL Anywhere Server - SQL Usage*].

Rebuilding version 9 and earlier databases for version 12

This section describes how to unload and reload your database into a new version 12 database.

For information about upgrading Windows Mobile databases, see "Rebuilding databases on Windows Mobile" [*SQL Anywhere Server - Database Administration*].

Rebuilding Mac OS X databases
SQL Anywhere 9.0.2 for Mac OS X was supported on PPC, while SQL Anywhere 10.0.0 and later for Mac OS X are supported on Intel. If you have a version 9.0.2 or earlier database on Mac OS X, you have two options for unloading the database:

- Unload the database using the version 9.0.2 software.
- Copy the database to a different platform where SQL Anywhere 12 is installed, and then unload the database using the version 12 software.

Once the database is unloaded, you can perform the reload on Mac OS X using the version 12 software.

Caution
Unloading and reloading a large database can be time consuming and can require a large amount of disk space. The process requires access to disk space approximately twice the size of your database to hold the unloaded data and the new database file.

Upgrade restrictions

There are some restrictions to note when rebuilding version 9 or earlier databases using the version 12 tools:

- You must disconnect the database from any earlier versions of the database server, and you must shut down any earlier database servers running on the computer. You must also shut down any version 12 database servers that are running on the computer. If dbunload cannot proceed because it detects any of these cases, it issues an error and fails.
- Do not include the ENG, START, or LINKS connection parameters in the dbunload connection string for the old database (specified in the -c option). If you specify these parameters, they are ignored and a warning appears. In the Sybase Central **Connect** window, do not enter values in the **Server Name** or **Start Line** fields.
- You must run dbunload on a computer with direct file system access to the old database (dbunload must be able to connect to the database using shared memory).
- You cannot run a database server named dbunload_support_engine on the computer where the rebuild is taking place.
- You cannot unload a version 9 or earlier database when the database file requires recovery. Recovery might be required, for example, if the database file was created using the Backup utility (dbbackup) or a BACKUP DATABASE statement. When you use the Unload utility (dbunload) on a database file that requires recovery, a message is returned indicating that the database could not be started. Use a version 9 database server to start the database and then stop the database before retrying dbunload.

Special considerations

- **Password case sensitivity** In newly-created SQL Anywhere 12 databases, all passwords are case sensitive, regardless of the case-sensitivity of the database. The default DBA password for new databases is **sql**.

  When you rebuild an existing database, SQL Anywhere determines the case sensitivity of the password as follows:

  - If the password was originally entered in a case-insensitive database, the password remains case-insensitive.
  - If the password was originally entered in a case-sensitive database, uppercase and mixed case passwords remain case sensitive. However, if the password was entered in all lowercase, then the password becomes *case-insensitive*.
  - Changes to both existing passwords and new passwords are case sensitive.

- **Page sizes** The default database page size for SQL Anywhere 12 databases is 4096 bytes. The supported page sizes in version 12 are 2048 bytes, 4096 bytes, 8192 bytes, 16384 bytes, and 32768 bytes. If your old database uses an unsupported page size, the new database has a page size of 4096 bytes by default. You can use the dbinit -p option or the dbunload -ap option to specify a different page size. See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*] and "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

- **Collations** In version 9 and earlier, SQL Anywhere supported one collation used with CHAR data types. This collation used the SQL Anywhere Collation Algorithm (SACA). In version 10 and later, SQL Anywhere supports two collation algorithms, SACA and UCA (Unicode Collation Algorithm). Unless you specify a new or different collation for the rebuilt database, the SACA collation from the old database is unloaded and reused in the rebuilt database.

  If you are rebuilding a database with a custom collation, the collation is preserved only if you rebuild in a single step (internal unload). If you choose to unload the database, and then load the schema and data into a database that you create, then you must use one of the supplied collations. See "Supported and alternate collations" [*SQL Anywhere Server - Database Administration*].

- **Database file size** Because of index changes in SQL Anywhere, when you rebuild a database by unloading and reloading it, the rebuilt database may be smaller than the original database. This decrease in database size does not indicate a problem or a loss of data.

Rebuild a version 9 or earlier database from Sybase Central

You can use the **Unload Database Wizard** to rebuild an old database. Use the wizard to unload into a reload file and data files, unload and reload into a new database, or unload and reload into an existing database. It is strongly recommended that you back up your database before rebuilding it.

Sybase Central upgrade notes

- The database file must be located on the same computer as the SQL Anywhere 12 installation.
- You cannot unload a subset of tables from a database. You must use the dbunload utility to do this.
- If the **Unload Database Wizard** determines that the database file is already running, then the database is stopped before the unload proceeds.

To rebuild a database (Sybase Central)

1. Carry out the standard precautions for upgrading software. See "Important upgrade precautions" on page 306.
2. If possible, defragment the drive where the new database will be stored because a fragmented drive can decrease database performance.
3. Ensure that you have exclusive access to the database to be unloaded and reloaded. No other users can be connected.
4. Choose **Start** » **Programs** » **SQL Anywhere 12** » **Administration Tools** » **Sybase Central**.
5. Choose **Tools** » **SQL Anywhere 12** » **Unload Database**.
6. Read the introductory page of the **Unload Database Wizard**, and click **Next**.
7. Select **Unload A Database Running On An Earlier Version Of The Server, Or A Database That Is Not Running**. Enter the connection information for the database. Click **Next**.
8. Select **Unload And Reload Into A New Database**. Click **Next**.
9. Specify a new file name for the database. Click **Next**.

   You can specify the page size for the new database. In version 12, the default (and recommended) page size is 4096 bytes.

   You can encrypt the database file if you want. If you choose strong encryption, you need the encryption key each time you want to start the database. See "Encrypting and decrypting a database" [*SQL Anywhere Server - Database Administration*].
10. Choose to unload structure and data. Click **Next**.
11. Specify whether you want to connect to the new database when the rebuild is complete.
12. Click **Finish**. Examine the new database to confirm that the rebuild completed properly.

Rebuild a version 9 or earlier database using the Unload utility

You can use the Unload utility (dbunload) -an or -ar option to rebuild an old database:

- The -an option is recommended because it creates a new database leaving the original database intact.

- The -ar option replaces your old database with a new version 12 database.

It is recommended that you back up your database before rebuilding it.

Note
The page size for a database can be (in bytes) 2048, 4096, 8192, 16384, or 32768, with the default being the page size of the original database.

To rebuild a database (command line)

1. Carry out the standard precautions for upgrading software. See "Important upgrade precautions" on page 306.
2. Ensure that the version 12 utilities are ahead of other utilities in your system path. See "Using the utilities" on page 305.
3. Shut down all SQL Anywhere and Adaptive Server Anywhere database servers because the version 12 dbunload utility cannot be used against a database that is running on a previous version of the database server. For example:

   ```
   dbstop -c "DBF=mydb.db;UID=DBA;PWD=sql"
   ```

4. If possible, defragment the drive where the new database will be stored because a fragmented drive can decrease database performance.
5. Back up the database. For example:

   ```
   dbbackup -c "DBF=mydb.db;UID=DBA;PWD=sql" old-db-backup-dir
   ```

   See "Backup quick start" [*SQL Anywhere Server - Database Administration*].
6. Run the Unload utility (dbunload) using the -an or -ar option to create a new database.

   ```
   dbunload -c "connection-string" -an database-filename
   ```

   For example:

   ```
   dbunload -c "DBF=mydb.db;UID=DBA;PWD=sql" -an mydb12.db
   ```

   The database user specified in the *connection-string* must connect to the database to be unloaded with DBA authority. This command creates a new database (by specifying -an). If you specify the -ar option, the existing database is replaced with a rebuilt database. To use the -ar option, you must connect to a personal database server or to a network database server on the same computer as the Unload utility (dbunload).

   For information about the other options available for the Unload utility (dbunload), see "Unload utility (dbunload)" [*SQL Anywhere Server - Database Administration*].

Known issues

If the rebuild process fails when you run dbunload or the **Unload Database Wizard**, you can use the following steps to help diagnose the reason for the failure.

To diagnose a rebuild failure

1. Run dbunload -n on your old database. Data is not unloaded because -n is specified.

   ```
   dbunload -c "connection-string" -n directory-name
   ```

2. Create a new, empty version 12 database.

   ```
   dbinit test.db
   ```

3. Apply the *reload.sql* file to the empty database.

   ```
   dbisql -c "DBF=test.db;UID=DBA;pwd=sql" reload.sql
   ```

4. Make changes to the *reload.sql* file or the original database based on the messages you receive when applying the *reload.sql* file to the new database.

The following table lists issues that are known to cause a rebuild to fail, as well as their solutions.

Known problem	Solution
A DECLARE LOCAL TEMPORARY TABLE statement in a procedure or trigger causes a syntax error if the table name is prefixed with an owner name.	Remove the owner name.
If a CREATE TRIGGER statement does not include an owner name for the table on which the trigger is defined, and the table must be qualified with an owner when referenced by the user executing the *reload.sql* file, the statement fails and an error is returned indicating that the table could not be found.	Prefix the table name with the owner name.

If an object name (such as a table, column, variable, or parameter name) corresponds to a reserved word introduced in a later version of SQL Anywhere (such as NCHAR), then the reload fails. For example:
`CREATE PROCEDURE p( )`
`BEGIN`
`DECLARE NCHAR INT;`
`SET NCHAR = 1;`
`END;`
	Change all references to the reserved word to use a different name. For variable names, prefixing the name with @ is a common convention that prevents naming conflicts.
	For a complete list of reserved words, see "Reserved words" [*SQL Anywhere Server - SQL Reference*].

Known problem	Solution
If a database is unloaded with a version 9 or earlier copy of dbunload, the *reload.sql* file can contain calls to the ml_add_property system procedure, but this procedure is not present in a new version 12 database.	Unload the database with the version 12 dbunload utility. For information about ensuring you are using the correct version of the database utilities, see "Using the utilities" on page 305.
If you unload a database using a version 9 or earlier version of dbunload, views that use Transact-SQL outer joins (by specifying *= or =*) may not be created properly when they are reloaded.	Add the following line to the reload script: `SET TEMPORARY OPTION tsql_outer_joins='on';` You should later rewrite any views that use Transact-SQL outer joins.
The [NOT] DETERMINISTIC clause is not supported in the CREATE PROCEDURE and ALTER PROCEDURE statements. If the clause is present, the reload fails and a syntax error is returned.	If you are upgrading a database that contains user-defined procedures that include the [NOT] DETERMINISTIC clause, you must remove the clause before you unload and reload the database.

Upgrading version 10 and later databases

Upgrading a database adds and modifies system tables, system procedures, and database options to enable version 12 features. It does not change the file format used to store and access data on disk, and so does not give access to all new features and performance enhancements in the latest version of the software.

For information about upgrading the database file format, see "Rebuilding version 10 and later databases" on page 307.

The **Upgrade Database Wizard** does not upgrade a version 9.0.2 or earlier database to version 12. To upgrade an existing version 9.0.2 or earlier database to version 12, you must unload and reload the database using dbunload or the **Unload Database Wizard**. See "Upgrading version 9 and earlier databases" on page 304.

Caution
You should always back up your database files before upgrading. If you apply the upgrade to the existing files, then these files become unusable if the upgrade fails. For information about backing up your database, see "Backup and data recovery" [*SQL Anywhere Server - Database Administration*].

To upgrade a database (Sybase Central)

1. Carry out the standard precautions for upgrading software. See "Important upgrade precautions" on page 306.

2. Choose **Start** » **Programs** » **SQL Anywhere 12** » **Administration Tools** » **Sybase Central**.

3. From the **SQL Anywhere 12** plug-in, connect to the database you want to upgrade. The database must be running on a version 12 database server.

4. Choose **Tools** » **SQL Anywhere 12** » **Upgrade Database**.

5. Follow the instructions in the **Upgrade Database Wizard**.

6. Stop the database and archive the transaction log by making a copy of it before using the upgraded database if you did not choose to do so in the wizard.

Tip
You can also access the **Upgrade Database Wizard** by:

- Right-clicking a database, and choosing **Upgrade Database**.
- Selecting a database, and choosing **File** » **Upgrade Database**.

To upgrade a database (command line)

1. Carry out the standard precautions for upgrading software. See "Important upgrade precautions" on page 306.

2. Ensure that you have exclusive access to the database to be upgraded and ensure that the version 12 utilities are ahead of other utilities in your system path. See "Using the utilities" on page 305.

3. Run the Upgrade utility (dbupgrad) against the database:

   ```
   dbupgrad -c "connection-string"
   ```

   The database user specified in the *connection-string* must have DBA authority on the database to be upgraded.

   For more information, see "Upgrade utility (dbupgrad)" [*SQL Anywhere Server - Database Administration*].

4. Shut down the database and archive the transaction log before using the upgraded database.

To upgrade a database (SQL)

1. Connect to the database from Interactive SQL or another application that can execute SQL statements. No other connection can be using the database at the same time.

2. Execute an ALTER DATABASE statement.

   For example, the following statement upgrades a database:

   ```
   ALTER DATABASE UPGRADE;
   ```


For more information, see "ALTER DATABASE statement" [*SQL Anywhere Server - SQL Reference*].

3. Shut down the database and archive the transaction log before using the upgraded database.

Upgrading SQL Anywhere software and databases in a database mirroring system

When you are using database mirroring, extra steps are required to apply a SQL Anywhere maintenance release or EBF, or to upgrade the database file:

- For information about applying a maintenance release, see "Installing SQL Anywhere maintenance releases in a database mirroring system" on page 317.
- For information about applying an EBF, see "Applying SQL Anywhere EBFs in a database mirroring system" on page 317.
- For information about upgrading or rebuilding the database file, see "Upgrading databases in a database mirroring system" on page 318.

Installing SQL Anywhere maintenance releases in a database mirroring system

All database servers in a database mirroring system must use the same maintenance release of SQL Anywhere. If you use the following procedure to apply a SQL Anywhere maintenance release, the only time the database is not available is during steps 3 and 4.

To apply a SQL Anywhere maintenance release to a database mirroring system

1. Shut down the mirror server by issuing a dbstop command.
2. Install the new version of SQL Anywhere on the mirror server.
3. Shut down the primary and arbiter servers by issuing a dbstop command for each server.
4. Install the new version of SQL Anywhere on the primary server.
5. Restart the primary and mirror servers.
6. Install the new version of the software on the arbiter.
7. Restart the arbiter.

Applying SQL Anywhere EBFs in a database mirroring system

To install an EBF, you must perform the following steps for each database server in the mirroring system (primary, mirror, and arbiter servers):

1. Issue a dbstop command to stop the database server.
2. Install the EBF.

3. Restart the database server.

The only downtime occurs during the failover caused by shutting down the primary server.

See also

- "Stopping a database server in a mirroring system" [*SQL Anywhere Server - Database Administration*]
- "Initiating failover on the primary server" [*SQL Anywhere Server - Database Administration*]

Upgrading databases in a database mirroring system

There are two procedures you can use to upgrade or rebuild a database that is participating in a database mirroring system. The first process is simpler, but it has a longer database downtime than the second procedure.

To upgrade or rebuild a database in a database mirroring system

1. Shut down the mirror server.
2. Shut down the primary server.
3. Upgrade or rebuild the database using the copy on the primary server. See "Upgrading version 10 and later databases" on page 315, or "Rebuilding version 10 and later databases" on page 307.
4. Copy the upgraded or rebuilt database and transaction log to the mirror server.
5. Restart the primary server.
6. Restart the mirror server.

> **Note**
> Any renamed transaction log files should be moved because they are incompatible with the new database. An initial transaction log file is required on both servers for mirroring to start. You can create a transaction log file by executing a dbping command against the database.

To minimize downtime while upgrading or rebuilding a database in a database mirroring system

1. Make a backup of the database and rename the transaction log.
2. Upgrade or rebuild the backup copy of the database on a different computer. See "Upgrading version 10 and later databases" on page 315 or "Rebuilding version 10 and later databases" on page 307.
3. Shut down both the primary and mirror servers.
4. Save the current copy of the transaction log on the primary database.
5. Use the dbtran utility to translate the transaction log saved in step 4.

   This transaction log contains any changes made to the database since the backup in step 1.

6. Start the rebuilt database using a local database server.
7. Apply the translated transaction log using the READ statement from Interactive SQL.
8. Stop the rebuilt database.
9. Copy the upgraded or rebuilt database and its transaction log to the primary and mirror servers.
10. Start the primary server.
11. Start the mirror server.

Applying SQL Anywhere EBFs in a read-only scale-out system

To apply an EBF in a read-only scale-out system, you shut down the database for the node and install the EBF. You do not have to install the EBF at the root node first.

Upgrading databases that use read-only scale-out

This procedure describes how to upgrade a database being used in a scale-out system, including when scale-out is being used with database mirroring.

To upgrade a database in a scale-out system

1. Make a backup of the database and rename the transaction log.
2. Upgrade or rebuild the backup copy of the database on a different computer. See "Upgrading version 10 and later databases" on page 315 or "Rebuilding version 10 and later databases" on page 307.
3. Stop the database servers for all of the copy nodes in the system.
4. Shut down the root server. If you are using database mirroring in conjunction with scale-out, you must shut down both the primary and mirror servers.
5. Save the current copy of the transaction log on the primary database.
6. Use the dbtran utility to translate the transaction log saved in step 4.

   This transaction log contains any changes made to the database since the backup in step 1.
7. Start the rebuilt database using a local database server.
8. Apply the translated transaction log using the READ statement from Interactive SQL.
9. Stop the rebuilt database.
10. Copy the upgraded or rebuilt database and its transaction log to the root database server.
11. Copy the upgraded database and an empty transaction log files to each copy server.
12. Start the root database server. If you are using database mirroring with scale-out, you must start both the primary and the mirror servers.

Upgrading MobiLink

Compatibility with existing software

- Version 12 MobiLink clients are incompatible with versions of the MobiLink server before 12.0.0.
- The version 12 MobiLink server can be used with clients that are version 10 or later. To use version 10 or 11 clients, start the MobiLink server with the -x option. If you need to support earlier clients, you should keep an earlier version of the MobiLink server to support them.
- Confirm that none of the documented behavior changes affect your application. If they do, you must update your application. See "SQL Anywhere 12 - Changes and Upgrading" on page 1.

Upgrade order

If you are upgrading an existing MobiLink installation, you must upgrade the components in the following order:

1. Shut down the MobiLink servers.
2. Upgrade the consolidated database.

   See "Upgrading your consolidated database" on page 320.
3. Upgrade the MobiLink servers.

   See "Upgrading the MobiLink server" on page 326.
4. Start the MobiLink servers.
5. Upgrade the MobiLink clients.

   The version 12 MobiLink server can only be used with clients that are version 10 or later. MobiLink clients prior to version 10 must be upgraded in order to work with version 12 MobiLink server.

   For information about SQL Anywhere remote databases, see "Upgrading SQL Anywhere MobiLink clients" on page 326. For information about UltraLite applications, see "Upgrading databases created with previous versions of UltraLite" on page 331.

Before upgrading, check for behavior changes that may affect you and take standard upgrade precautions.

See also

- "Behavior changes and deprecated features" on page 263
- "Important upgrade precautions" on page 306

Upgrading your consolidated database

Before you can use the new MobiLink server with an existing consolidated database, you must run upgrade scripts that install new system objects. The upgrade scripts must be run by the owner of the

currently installed MobiLink system tables. You can also use the following methods to update the MobiLink system setup:

- In the MobiLink plug-in for Sybase Central, choose **MobiLink 12** » **Project** » **Consolidated Databases** and right-click the database name and choose **Check MobiLink System Setup**. If your database requires setup, you are prompted to continue.

- When you use the **Deploy Synchronization Model Wizard**, system setup is checked when you connect to your server database. If your database requires setup, you are prompted to continue. See "Introduction to synchronization models" [*MobiLink - Getting Started*].

The MobiLink upgrade scripts for 6.0.x have been removed. If you require this upgrade, contact Technical Support (http://www.sybase.com/support).

Notes

- Use the ml_add_missing_dnld_scripts stored procedure to fix missing download_cursor and/or download_delete_cursor scripts. Invoking this procedure with a script version name defines the missing download_cursor and/or download_delete_cursor scripts as ignored scripts for every synchronization table used by the given script version.

- If you have authenticate_user_hashed scripts that were created earlier than version 10.0.0, you must change them to accept VARBINARY(32) instead of VARBINARY(20), using the binary equivalent type of your RDBMS.

Upgrading SQL Anywhere version 10.0.0 and later

To upgrade a consolidated database (SQL Anywhere 10.0.0 and later)

1. Upgrade the SQL Anywhere database.

   See "Upgrading version 10 and later databases" on page 304.

2. Run the appropriate upgrade script for the version you are upgrading from.

   The upgrade script is called *upgrade_sa.sql*. It is located under your SQL Anywhere installation in *MobiLink\upgrade\version*, where *version* is the SQL Anywhere version you are upgrading from.

   For example, connect to the database in Interactive SQL and run the following command:

   ```
   READ "C:\Program Files\SQL Anywhere 12\MobiLink\upgrade
   \10.0.1\upgrade_sa.sql"
   ```

Upgrading earlier versions of SQL Anywhere

- Before SQL Anywhere version 10.0.0, MobiLink system tables were owned by dbo. To run the setup scripts for a SQL Anywhere database, you must be logged in to the consolidated database as the owner of the MobiLink system tables. It is not enough to run these scripts as a user with permission to change the tables. To run the upgrade scripts, you can use the SETUSER SQL statement to impersonate dbo. For example:

  ```
  SETUSER "dbo";
  ```

To upgrade a consolidated database in Sybase Central, you should use the GRANT CONNECT statement to create a password for dbo and then connect as dbo. For example:

```
GRANT CONNECT TO dbo IDENTIFIED BY password;
```

In the latter case, after you have upgraded you should use ALTER USER to remove the dbo password. For example:

```
ALTER USER TO dbo IDENTIFIED BY "";
```

- If you have set up a SQL Anywhere consolidated database but never synchronized with it, then you must run the setup script (not the upgrade script). This step only applies to SQL Anywhere consolidated databases. See "SQL Anywhere consolidated database" [*MobiLink - Server Administration*].

To upgrade a consolidated database (SQL Anywhere earlier than 10.0.0)

1. If you are upgrading a SQL Anywhere consolidated database that is earlier than version 10.0.0, you must first upgrade the database to version 12:
   a. Shut down the database server.
   b. Upgrade the database to version 12.

      For instructions, see:

      - "Upgrading version 10 and later databases" on page 315
      - "Rebuilding version 10 and later databases" on page 307
      - "Rebuilding version 9 and earlier databases for version 12" on page 309

   c. Start the database server, logging in as DBA.

      You must log in as DBA to upgrade.

2. Run the appropriate upgrade script for the version you are upgrading from.

   The upgrade script is called *upgrade_asa.sql*. It is located under your SQL Anywhere installation in *MobiLink\upgrade\version*, where *version* is the SQL Anywhere version you are upgrading from.

   To run the upgrade scripts, you must impersonate the dbo user. You can do this with the SETUSER SQL statement.

   For example, to upgrade a SQL Anywhere version 9.0.2 consolidated database, connect to the database in Interactive SQL and run the following command:

   ```
   SETUSER "dbo";
   READ 'C:\Program Files\SQL Anywhere 12\MobiLink\upgrade
   \9.0.2\upgrade_asa.sql'
   ```

3. Remove the dbo password. For example:

   ```
   GRANT CONNECT TO "dbo";
   ```

4. If you are running the MobiLink server as a user other than DBA, you must grant execute permission for that user on the new MobiLink system objects. Which system objects are new depends on which version you are upgrading from. The following code grants the necessary permissions to all MobiLink

system objects. Before executing the code, you must change the user name my_user to the name of the user who is running the MobiLink server.

```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_column to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_connection_script to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_database to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_device to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_device_address to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_listening to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough_repair to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough_script to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_passthrough_status to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_primary_server to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_property to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_delivery to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_delivery_archive to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_global_props to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_notifications to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_archive to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_props to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_props_archive
to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_repository_staging to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_status_history to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_status_history_archive
to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_qa_status_staging to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_agent to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_agent_property to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_agent_staging to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_deployed_task to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_event to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_event_staging to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_managed_remote to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_notify to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_remote_db_class to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_task to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_task_command to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_task_command_property to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_ra_task_property to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_script to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_script_version to my_user;
```

```
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_scripts_modified to
my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_sis_sync_state to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_subscription to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_table to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_table_script to my_user;
GRANT SELECT, INSERT, UPDATE, DELETE ON dbo.ml_user to my_user;
GRANT EXECUTE ON dbo.ml_add_column to my_user;
GRANT EXECUTE ON dbo.ml_add_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_dnet_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_dnet_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_java_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_java_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_conn_script_chk to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_connection_script to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_lang_table_script_chk to my_user;
GRANT EXECUTE ON dbo.ml_add_passthrough to my_user;
GRANT EXECUTE ON dbo.ml_add_passthrough_repair to my_user;
GRANT EXECUTE ON dbo.ml_add_passthrough_script to my_user;
GRANT EXECUTE ON dbo.ml_add_property to my_user;
GRANT EXECUTE ON dbo.ml_add_table_script to my_user;
GRANT EXECUTE ON dbo.ml_add_user to my_user;
GRANT EXECUTE ON dbo.ml_delete_device to my_user;
GRANT EXECUTE ON dbo.ml_delete_device_address to my_user;
GRANT EXECUTE ON dbo.ml_delete_listening to my_user;
GRANT EXECUTE ON dbo.ml_delete_passthrough to my_user;
GRANT EXECUTE ON dbo.ml_delete_passthrough_repair to my_user;
GRANT EXECUTE ON dbo.ml_delete_passthrough_script to my_user;
GRANT EXECUTE ON dbo.ml_delete_remote_id to my_user;
GRANT EXECUTE ON dbo.ml_delete_sync_state to my_user;
GRANT EXECUTE ON dbo.ml_delete_sync_state_before to my_user;
GRANT EXECUTE ON dbo.ml_delete_user to my_user;
GRANT EXECUTE ON dbo.ml_delete_user_state to my_user;
GRANT EXECUTE ON dbo.ml_lock_rid to my_user;
GRANT EXECUTE ON dbo.ml_qa_add_delivery to my_user;
GRANT EXECUTE ON dbo.ml_qa_add_message to my_user;
GRANT EXECUTE ON dbo.ml_qa_handle_error to my_user;
GRANT EXECUTE ON dbo.ml_qa_stage_status_from_client to my_user;
GRANT EXECUTE ON dbo.ml_qa_staged_status_for_client to my_user;
GRANT EXECUTE ON dbo.ml_qa_upsert_global_prop to my_user;
GRANT EXECUTE ON dbo.ml_ra_add_agent_id to my_user;
GRANT EXECUTE ON dbo.ml_ra_add_managed_remote_id to my_user;
GRANT EXECUTE ON dbo.ml_ra_assign_task to my_user;
GRANT EXECUTE ON dbo.ml_ra_cancel_notification to my_user;
GRANT EXECUTE ON dbo.ml_ra_cancel_task_instance to my_user;
GRANT EXECUTE ON dbo.ml_ra_clone_agent_properties to my_user;
GRANT EXECUTE ON dbo.ml_ra_delete_agent_id to my_user;
GRANT EXECUTE ON dbo.ml_ra_delete_events_before to my_user;
GRANT EXECUTE ON dbo.ml_ra_delete_remote_id to my_user;
GRANT EXECUTE ON dbo.ml_ra_delete_task to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_agent_events to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_agent_ids to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_agent_properties to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_latest_event_id to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_orphan_taskdbs to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_remote_ids to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_task_results to my_user;
GRANT EXECUTE ON dbo.ml_ra_get_task_status to my_user;
GRANT EXECUTE ON dbo.ml_ra_int_cancel_notification to my_user;
GRANT EXECUTE ON dbo.ml_ra_int_move_events to my_user;
GRANT EXECUTE ON dbo.ml_ra_notify_agent_sync to my_user;
GRANT EXECUTE ON dbo.ml_ra_notify_task to my_user;
GRANT EXECUTE ON dbo.ml_ra_reassign_taskdb to my_user;
```



```
GRANT EXECUTE ON dbo.ml_ra_set_agent_property to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_agent_auth_file_xfer to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_download_ack to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_download_prop to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_download_remote_dbs to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_download_task to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_download_task_cmd to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_end_upload to my_user;
GRANT EXECUTE ON dbo.ml_ra_ss_upload_property to my_user;
GRANT EXECUTE ON dbo.ml_ra_unmanage_remote_id to my_user;
GRANT EXECUTE ON dbo.ml_reset_sync_state to my_user;
GRANT EXECUTE ON dbo.ml_set_device to my_user;
GRANT EXECUTE ON dbo.ml_set_device_address to my_user;
GRANT EXECUTE ON dbo.ml_set_listening to my_user;
GRANT EXECUTE ON dbo.ml_set_sis_sync_state to my_user;
GRANT EXECUTE ON dbo.ml_upload_update_device_address to my_user;
GRANT EXECUTE ON dbo.ml_upload_update_listening to my_user;
```

Upgrading Adaptive Server Enterprise, Oracle, MySQL, or Microsoft SQL Server MobiLink system objects

You only need to upgrade the MobiLink system objects in your Adaptive Server Enterprise, Oracle, MySQL, or Microsoft SQL Server consolidated database if your version of the MobiLink server is earlier than version 12.0.0.

To upgrade a consolidated database (Adaptive Server Enterprise, Oracle, MySQL, or Microsoft SQL Server)

1. For Adaptive Server Enterprise databases, you must set "select into" permission. Run the following command in Sybase Interactive SQL:

   ```
   USE MASTER
   go
   sp_dboption your-database-name, "SELECT INTO", true
   go
   USE your-database-name
   go
   checkpoint
   go
   ```

2. Run the appropriate upgrade script for the version you are upgrading from.

   The upgrade scripts are called *upgrade_XXX.sql*, where *XXX* indicates the RDBMS of your consolidated database. They are located under your SQL Anywhere installation in *MobiLink\upgrade\version*, where *version* is the MobiLink version you are upgrading from.

   For example, to upgrade a Microsoft SQL Server version 9.0.2 consolidated database, run the following command:

   ```
   osql -S server_name -U user_name -P password -I
   "C:\Program Files\SQL Anywhere 12\MobiLink\upgrade\9.0.2\upgrade_mss.sql"
   ```

Upgrading IBM DB2 LUW

You only need to upgrade your IBM DB2 LUW consolidated database if your version of the MobiLink server is earlier than version 12.0.0.

For information about how to run the IBM DB2 LUW setup script, see "IBM DB2 LUW consolidated database" [*MobiLink - Server Administration*].

To upgrade a IBM DB2 LUW consolidated database

1. Locate the IBM DB2 LUW upgrade script.

   The upgrade script is called *upgrade_db2.sql* and is held in the *MobiLink/upgrade/version* subdirectory of your SQL Anywhere installation. The *version* directory refers to the version of MobiLink from which you are upgrading.

2. Copy *upgrade_db2.sql* and modify the copy. Change the CONNECT statement at the start of the script so it works with the instance you want to connect to. Apply the copied SQL script to the consolidated database.

Upgrading the MobiLink server

You need to upgrade the MobiLink server to version 12 if you are using version 12 remotes.

Before using a version 12 MobiLink server, check the behavior changes to see if any affect you. See "SQL Anywhere 12 - Changes and Upgrading" on page 1.

Version 12 of the MobiLink server only supports version 10, 11 and 12 SQL Anywhere and UltraLite clients. If you need to support earlier clients, you should keep an earlier version of the MobiLink server for supporting them.

Upgrading SQL Anywhere MobiLink clients

In a production environment, only upgrade SQL Anywhere remote databases after you have upgraded both the consolidated database and the MobiLink server.

In version 10.0.0, Adaptive Server Anywhere was renamed to SQL Anywhere.

There are several kinds of upgrade to consider:

- Upgrading the software.
- Upgrading the remote database itself.
- Upgrading the whole application.

Upgrading the software

It is recommended that you upgrade dbmlsync and the SQL Anywhere database server at the same time. You must upgrade the remote database before running the new dbmlsync utility.

Version 12 MobiLink clients require a version 12 MobiLink server for synchronization. Version 12 MobiLink clients do not synchronize with a MobiLink server earlier than version 12.

For information about upgrading MobiLink, see "Upgrading MobiLink" on page 320.

Upgrading the remote database

You can upgrade MobiLink SQL Anywhere remote databases using the procedures for SQL Anywhere databases. For instructions, see "Upgrading SQL Anywhere" on page 304.

When there is a schema change or other significant database change you may need to perform a manual unload and reload.

To unload/reload a remote SQL Anywhere database manually

1. Stop all database activity.
2. Perform a successful synchronization and validate and back up the remote database.
3. Run the dbtran utility to display the starting offset and ending offset of the database transaction log. Make note of the ending offset.

   See "Log Translation utility (dbtran)" [*SQL Anywhere Server - Database Administration*].
4. Rename the transaction log. This ensures that it is not modified during the unload process. Move the renamed log file to a secure location, such as an offline directory.
5. Unload the database.

   See "Rebuilding version 9 and earlier databases for version 12" on page 309.
6. Initialize a new database.

   See "Initialization utility (dbinit)" [*SQL Anywhere Server - Database Administration*].
7. Reload the data into the new database.

   See "Rebuilding version 9 and earlier databases for version 12" on page 309.
8. Shut down the new database.
9. Erase the new database's transaction log.
10. Run dblog on the new database, using the following options:
    - Use -z to specify the ending offset that you noted earlier.
    - Use -x to set the relative offset to zero.

    For example:

    ```
    dblog -x 0 -z 137829 database-name.db
    ```

    See "Transaction Log utility (dblog)" [*SQL Anywhere Server - Database Administration*].
11. Start dbmlsync, specifying the location of the original log file that you moved earlier.

See "dbmlsync syntax" [*MobiLink - Client Administration*].

12. When you no longer need the old log file, set the database option delete_old_logs.

    See "delete_old_logs option [SQL Remote]" [*SQL Anywhere Server - Database Administration*].

Upgrading applications

When deploying a new version of a MobiLink application, it is recommended that you use a new version name for the synchronization scripts. For example, if the existing application uses a script version called **v1**, then the upgraded application could use a script version called **v2**. Both script versions can be in use at the same time. This makes it easier to upgrade the remote databases incrementally, rather than all at once.

For version 9.0.0 and later, the MobiLink server -zd option has been removed. If your deployment uses the -zd option and you want to upgrade, you must change your download scripts to accept the last download timestamp as the first parameter. Alternatively, you can upgrade your client and start using named parameters, which enable you to put script parameters in any order.

Upgrading when using the SQL Anywhere Monitor for MobiLink

You can upgrade the SQL Anywhere Monitor at any time before or after upgrading the MobiLink servers.

To avoid unwanted alerts, you should schedule a blackout period in the SQL Anywhere Monitor for your servers before you shut them down.

Upgrading QAnywhere

To upgrade a QAnywhere application, you can upgrade your consolidated database, application, and client message stores.

To upgrade the consolidated database, see "Upgrading your consolidated database" on page 320.

To upgrade your applications, you should review the new features and behavior changes in this release.

See "SQL Anywhere 12 - Changes and Upgrading" on page 1.

To upgrade QAnywhere message stores

1. Deploy QAnywhere files.

   See "Deploying QAnywhere applications" [*MobiLink - Server Administration*].

2. Upgrade the message store:

Run the QAnywhere Agent with the -su option or the -sur option. See:

- "-su qaagent option" [*QAnywhere*]
- "-sur qaagent option" [*QAnywhere*]

Upgrading QAnywhere UltraLite and standalone clients

Upgrading from a pre-12.0.0 message store is performed by erasing the message store and then recreating it with the 12.0.0 utilities. During the first synchronization of the new message store, all client properties and unacknowledged messages will be propagated to the message store.

To upgrade a QAnywhere UltraLite client

1. Perform a synchronization of the message store immediately before beginning upgrading the procedure.

   Caution
   Any messages or properties created but not synchronized to the server before upgrading will be lost.

2. Delete the message store by deleting the corresponding UltraLite database: `del qanywhere.udb`
3. Recreate and reinitialize the message store with the 12.0.0 utilities. See "Setting up the client message store" [*QAnywhere*].
4. Perform an initial synchronization to repopulate the message store.

To upgrade a QAnywhere standalone client

1. Perform a synchronization of the message store immediately before beginning upgrading the procedure.

   Caution
   Any messages or properties created but not synchronized to the server before upgrading will be lost.

2. Delete the message store by deleting the corresponding UltraLite database: `del qanywhere.udb`
3. The QAnywhere client application recreates the message store automatically next time it is started.
4. Perform an initial synchronization to repopulate the message store.

Upgrading UltraLite

Before using existing applications with this version of the software, be sure to review the list of new features and behavior changes to determine whether your application is affected. See:

- "UltraLite new features" on page 143
- "UltraLite behavior changes and deprecated features" on page 149

Using UltraLite utilities

If you have multiple versions of SQL Anywhere on your computer, you must pay attention to your system path when using UltraLite utilities to make sure that you are using the version 12 utilities. See "Using the utilities" on page 305.

UltraLite version 12

UltraLite's file format changed between versions 9 and 10 and then again between versions 11 and 12. UltraLite version 12 cannot read UltraLite databases created using any prior version of the software.

Upgrading version 10 and 11 databases

For version 10 and 11 UltraLite databases, use the version 11 unloadold utility to generate an XML file that can then be used by ulload to create a version 12 database. Once the database is upgraded, it cannot connect to version 10 or 11 applications, utilities, and software. See "Upgrading databases created with previous versions of UltraLite" on page 331.

Upgrading version 9 and earlier databases

For version 9 and earlier UltraLite databases, use the version 9 unloadold utility to generate an XML file that can then be used by ulload to create a version 12 database. Attempting to connect version 9 or earlier databases with UltraLite 12 results in an error on database startup. See "Upgrading databases created with previous versions of UltraLite" on page 331.

Compatibility with existing software

- UltraLite for AppForge is not supported by version 12.
- UltraLite 12 database files only support connections from version 12 client applications or the version 12 UltraLite engine.
- UltraLite 12 automatically upgrades version 10 database files.
- The UltraLite version 12 runtime and the UltraLite version 12 engine do not work with database files and application code created with version 9 and earlier of UltraLite.
- Management of old databases and client applications from the current version of Sybase Central is provided as follows:
  - Full management of version 12 databases.
  - Full management of version 10 databases. They are automatically upgraded to version 12.
  - You can only connect to a version 8 or 9 database to upgrade the database file format.

Palm OS
The Palm OS is not supported as of UltraLite version 12.

Upgrading databases created with previous versions of UltraLite

UltraLite's file format changed between versions 9 and 10 and then again between versions 11 and 12. UltraLite version 12 cannot read UltraLite databases created using any prior version of the software.

Notes

- Databases can only be upgraded on the desktop
- Make a backup copy of your database
- Synchronize your database if it is a production database that may contain unsynchronized changes

Upgrading UltraLite database on Windows

To upgrade UltraLite database on Windows

1. If the database was created with version 9 or earlier versions of UltraLite, open a command prompt and go to your *install-dir\UltraLite\Unload\V9*.

   If the database was created with version 10 or 11 of UltraLite, open a command prompt and go to *install-dir\UltraLite\Unload\V11*.

2. Run the ulunload utility to create an XML file (or files) with the contents of the database.

3. Use the ulload command line utility, or use the **Load** wizard in the UltraLite plug-in for Sybase Central to load the schema and data into a new version 12 database. Note that UltraLite databases now default to UTF8 encoding. If this does not suit your needs, you may need to explicitly set the utf8_encoding parameter to "off."

4. Check the generated XML file to verify the setting of the UTF-8 encoding. See "UltraLite utf8_encoding creation parameter" [*UltraLite - Database Management and Reference*].

To upgrade UltraLite database on Linux

1. Open a command prompt and go to *install-dir\ultraLite\unload\v11*.

2. Run the ulunload utility to create an XML file (or files) with the contents of the database.

3. Use the ulload command line utility, or use the **Load** wizard in the UltraLite plug-in for Sybase Central to load the schema and data into a new version 12 database. Note that UltraLite databases now default to UTF8 encoding. If this does not suit your needs, you may need to explicitly set the utf8_encoding parameter to "off."

4. Check the generated XML file to verify the setting of the UTF-8 encoding. See "UltraLite utf8_encoding creation parameter" [*UltraLite - Database Management and Reference*].

Upgrading SQL Remote

SQL Remote installations include a consolidated database and many remote databases, together with a SQL Remote Message Agent at each site.

At each site, the SQL Remote Message Agent handles the sending and receiving of messages. The messages take the form of SQL statements, and the database server handles the actual execution of those SQL statements.

The upgrade requirements for SQL Remote are as follows:

- **Software upgrades can be one site at a time** Older Message Agents (dbremote) can exchange messages with version 12 Message Agents. For version 5 of SQL Remote, the version 5 Message Agents can exchange messages with version 12 Message Agents, as long as the compression database option is set to a value of -1. There is no need to upgrade software throughout the installation simultaneously. See "compression option [SQL Remote]" [*SQL Anywhere Server - Database Administration*].

- **Upgrade databases** If you are upgrading a remote or consolidated database that used SQL Anywhere version 9 or lower, you must upgrade the database file format by unloading and reloading your database. There is no need for all databases to be upgraded at the same time.

  For instructions on unloading and reloading the database, see "Rebuilding version 9 and earlier databases for version 12" on page 309.

- **Upgrading Adaptive Server Enterprise consolidated databases** SQL Remote no longer supports Adaptive Server Enterprise consolidated databases. To synchronize Adaptive Server Enterprise databases, you should upgrade to MobiLink.

  For information about migrating from SQL Remote to MobiLink, see http://www.sybase.com/detail?id=1034174#335.

- **Upgrading with SQL Remote** Support for the VIM and MAPI message systems for SQL Remote was removed in version 11.0.0. When you upgrade a database that uses VIM or MAPI to SQL Anywhere version 12, you must change the message type to File, FTP, or SMTP. If the message type is MAPI or VIM, *dbremote.exe* does not start.

Example

One approach to upgrading version 5 of SQL Remote is as follows:

1. Upgrade the consolidated database server and SQL Remote Message Agent, and then upgrade the database file by unloading and reloading the consolidated database. Set the compression database option to -1, so that all messages are compatible with the version 5 software at remote sites. To unload and reload the consolidated database, see "Rebuild databases involved in synchronization or replication" [*SQL Anywhere Server - SQL Usage*].

2. One at a time, upgrade the remote database servers and Message Agents, and then upgrade the database file format by unloading and reloading the remote databases. You can set the compression database option to a value other than -1 to take advantage of compression and encoding on messages

being sent to the consolidated database server. To unload and reload the remote databases, see "Rebuild databases involved in synchronization or replication" [*SQL Anywhere Server - SQL Usage*]..

3. When all remote database servers and Message Agents are upgraded, set the compression database option at the consolidated site to a value other than -1.

Upgrading the SQL Anywhere Monitor and migrating resources and metrics

Caution
Uninstalling the Monitor removes the application, as well as resources and collected metrics.

If you want to preserve your current Monitor resources and metrics, you must:

1. Install a new version of the Monitor.
2. Migrate the resources and metrics.
3. Uninstall the older version of the Monitor.

You can use the Migrator utility to migrate the resources and metrics from one Monitor to a *newly* created Monitor.

To migrate the Monitor resources and data using the Migrator utility

1. Copy the existing Monitor database file, *samonitor.db*. For example to migrate the 11.0.1 Monitor, copy the following file:

   ```
   C:\Documents and Settings\All Users\Documents\SQL Anywhere 11\Monitor
   \samonitor.db
   ```

2. Install the new Monitor. Run the *setup.exe* file from the *Monitor* directory on your installation media, and follow the instructions provided. When the installation finishes, stop the new Monitor (if it is running).

   Note
   Only one version of the Monitor can run on a computer at a time.

   When you install the version 12.0.0 Monitor on a computer where a version 11.0.1 Monitor is running, the install stops the version 11.0.1 Monitor.

3. At a command prompt, run the Migrator utility. Use the following table to determine the location of the Migrator utility.

Operating system	Monitor type	Default location of the Migrator utility.
Windows	Monitor Developer Edition	*C:\Program Files\SQL Anywhere 12\run_migrator.cmd*
	Monitor Production Edition	*C:\Program Files\SQL Anywhere Monitor 12\run_migrator.cmd*
Linux	Monitor Developer Edition	*/opt/sqlanywhere12/bin32/run_migrator.sh*
	Monitor Production Edition	*/opt/samonitor12/bin32/run_migrator.sh* or */opt/samonitor12/bin64/run_migrator.sh*

Run the Migrator utility with the following options:

- **-c** Specifies that only resources and configuration settings are migrated. By default, both the resources and collected data are migrated.
- **-t temporary-directory** Specifies the directory for temporary files. By default, the temporary files are created in the same directory as the run_migrator file.

  Note
  The Monitor Migrator creates temporary files that are deleted at the end of the migration process. Use the -t option to specify a directory for these temporary files. The temporary files take up a similar amount of space as the version 11.0.1 Monitor database file. Ensure that the specified directory has sufficient space.

- **source-filename** Specifies the path and file name to the old Monitor file from which the data is unloaded. For example, the path to the version 11.0.1 *samonitor.db* file.
- ***destination--filename*** Specifies the path and file name to the new Monitor file where the data is reloaded. For example, the path to the version 12.0.0 *samonitor.db* file.

For example:

```
C:\Program Files\SQL Anywhere 12\run_migrator.cmd -t c:\monitorbackup c:
\Program Files\SQL Anywhere 11\Monitor\samonitor11.db C:\Program Files\SQL
Anywhere 12\Monitor\samonitor12.db
```

The default locations of the version 11.0.1 Monitor and version 12.0.0 Monitor database files are listed in the following table:

Operating system	Monitor type	11.0.1 directory	12.0.0 directory
Windows XP	Monitor Developer Edition	C:\Documents and Settings\All Users\Documents\SQL Anywhere 11\Monitor\samonitor.db	C:\Documents and Settings\All Users\Documents\SQL Anywhere 12\Monitor\samonitor.db
Windows XP	Monitor Production Edition	C:\Documents and Settings\All Users\Documents\SQL Anywhere 11 Monitor\samonitor.db	C:\Documents and Settings\All Users\Documents\SQL Anywhere 12 Monitor\samonitor.db
Windows Vista and later versions of Windows	Monitor Developer Edition	C:\Users\Public\Documents\SQL Anywhere 11\Monitor\samonitor.db	C:\Users\Public\Documents\SQL Anywhere 12\Monitor\samonitor.db
Windows Vista and later versions of Windows	Monitor Production Edition	C:\Users\Public\Documents\SQL Anywhere 11 Monitor\samonitor.db	C:\Users\Public\Documents\SQL Anywhere 12 Monitor\samonitor.db
Linux	Monitor Developer Edition	/opt/sqlanywhere11/samonitor.db	/opt/sqlanywhere12/samonitor.db
Linux	Monitor Production Edition	/opt/samonitor11/samonitor.db	/opt/sqlanywhere12/samonitor.db

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

