CHAPTER 2

Database Terms and Concepts

About this chapter

This chapter describes some basic relational database terms and concepts, which you need to work with SQL Anywhere Studio.

Contents

Database computing concepts

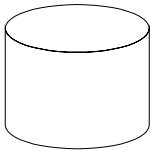
This section describes how database applications and the database server work together to manage databases.

Database computing components

Any information system contains the following pieces:

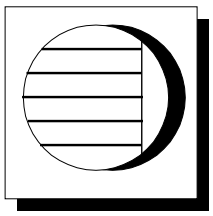
- **A database** Data is stored in a database. An Adaptive Server Anywhere database is a file, usually with an extension of *db*. SQL Anywhere Studio includes a sample database for you to work with: this is the file *asademo.db* in your Adaptive Server Anywhere installation directory.

  In diagrams in the documentation, a database is indicated by a cylinder:

- **A database server** The database server manages the database. No other applications address the database file directly; they all communicate with the database server.

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

  Adaptive Server Anywhere provides two versions of its database server: the **personal database server** and the **network database server**. In addition to the features of the personal server, the network server supports client/server communications across a network. The request-processing engine is identical in the two servers.

  For more information about these two versions of the server, see "The Adaptive Server Anywhere database server" on page 13.

- **A language interface** Applications communicate with the database server using an interface. You can use ODBC, JDBC, Sybase Open Client, or Embedded SQL.

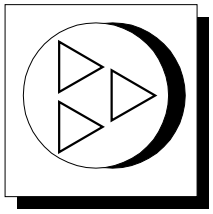
  The language interface provides a set of function calls for communicating with the database. For ODBC and JDBC, the library is commonly called a **driver**. The interface is typically provided as a shared library on UNIX operating systems or a dynamic link library (DLL) on PC operating systems. The JDBC interface uses the Sybase jConnect driver, which is a zip file of compiled Java classes.

  If you are working with an Adaptive Server Anywhere network server, the language interface resides on the client computer.

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

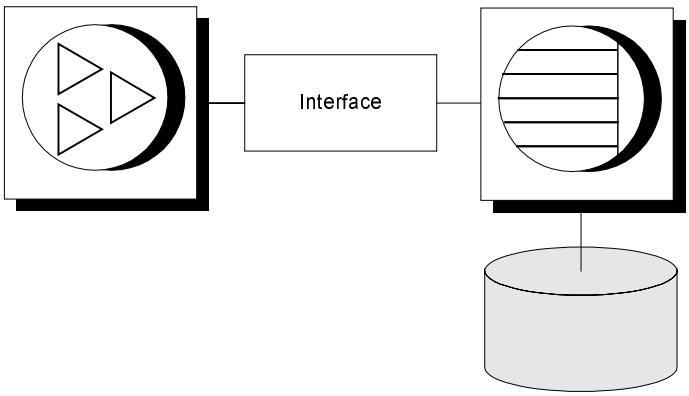
  If you develop an application using a rapid application development (RAD) tool such as one of the Powersoft PowerStudio tools, you may find that the tool provides its own methods for communicating with database servers, and hide the details of the language interface. Nevertheless, all applications do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

  In diagrams in the documentation, a client application is indicated by the following:



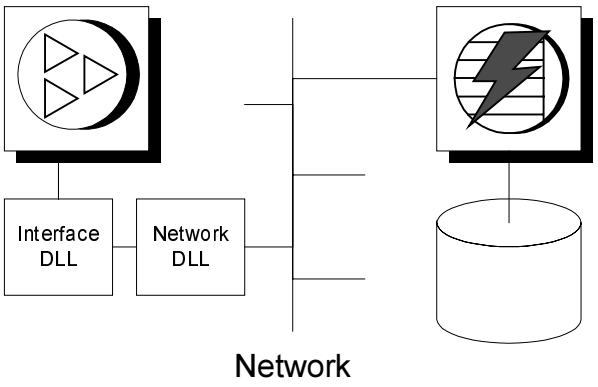
How the pieces fit together

A complete application on a single computer has the following architecture:



In this case the database server is generally the personal server, although you can also use applications on the same machine as the network server.

A complete application in a client/server environment has the following architecture:



In this case the database server is the network server, which supports network communications. No changes are needed to the client application for it to work in a client/server environment.

SQL and database computing

When a client application wants to carry out a database task, such as retrieving information using a query or inserting a new row into a table, it does so using Structured Query Language (SQL) statements. SQL, pronounced "sequel", is a relational database language that has been standardized by the ANSI and ISO standards bodies.

Depending on how you develop a client application, SQL statements could be supplied in function calls from the programming language, or you may build them graphically in a special window provided by your application development tool.

The programming interface delivers the SQL statement to the database server. The database server receives the statement and executes it, returning any required information (such as query results) back to the application.

Client/server communications protocols carry information between the client application and the database server, and programming interfaces define how an application sends the information. No matter what interface you use, and what network protocol you use, it is SQL statements that are sent to a server, and the results of SQL statements that are returned to the client application.

For an introduction to SQL see the chapters beginning with "Selecting Data from Database Tables" on page 231.

Other files used by Adaptive Server Anywhere

In addition to the database file, Adaptive Server Anywhere uses two other files when it is running a database. These are the transaction log and the temporary file.

The transaction log

The transaction log is a separate file, which contains a record of all the operations performed on the database. Normally, it has the same name as the database file, except that it ends with the suffix *.log* instead of *.db*. It has three important functions.

- **Record operations on your data to enable recovery** You can recreate your database from the transaction log if the database file is damaged.
- **Enable database replication** SQL Remote can use this file to replicate your database on portable computers which are sometimes, but not always, connected to the network.
- **Improve performance** By writing information to the transaction log, the database server can safely process your commands without writing to the database file as frequently.

The temporary file

The temporary file is opened when the database server starts, and is closed down when the server stops. As its name suggests, the temporary file is used while the server is running to hold temporary information. The temporary file holds no information that needs to be kept between sessions.

The temporary file is stored in your temporary directory. The location of this directory is generally identified by your TEMP environment variable.

Relational database concepts

A relational database management system (RDBMS) is a system for storing and retrieving data, in which the data is represented in tables. A relational database consists of a collection of tables that store interrelated data.

This section introduces some of the terms and concepts that are important in talking about relational databases.

Database tables

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

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

A typical fragment of a table containing employee information may look as follows:

emp_ID	emp_lname	emp_fname	emp_phone
10057	Huong	Zhang	1096
10693	Donaldson	Anne	7821

Characteristics of relational tables

The tables of a relational database have some important characteristics:

- There is no significance to the order of the columns or rows
- Each row contains one and only one value for each column
- Each value for a given column has the same type

The following table lists some of the formal and informal relational database terms describing tables and their contents, together with their equivalent in other nonrelational databases. This manual uses the informal terms.

Formal relational term	Informal relational term	Equivalent nonrelational term
Relation	Table	File
Attribute	Column	Field
Tuple	Row	Record

What do you keep in each table?

When you are designing your database, you should make sure that each table in the database holds information about a specific thing, such as employees, products, or customers.

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

A relational database is not a set of unrelated tables. You use primary and foreign keys to describe relationships between the information in different tables.

Primary and foreign keys

Primary and foreign keys enable each row in the database tables to be identified, and define the relationships between the tables to be defined. These keys define the relational structure of a database.

Tables have a primary key

All tables in a relational database should have a **primary key**. The primary key is a column, or set of columns, that allows each row in the table to be uniquely identified. No two rows in a table with a primary key may have the same value of a primary key.

If no primary key is assigned, all the columns together become the primary key. It is good practice to keep your primary key for each table as compact as possible.

Examples

In a table holding information about employees, the primary key may be an ID number assigned to each employee.

In the sample database, the table of sales order items has the following columns:

- An order number, identifying the order the item is part of.
- A line number, identifying each item on any order.
- A product ID, identifying the product being ordered.
- A quantity, showing how many items were ordered.
- A ship date, showing when the order was shipped.

In order to identify a particular item, both the order number and the line number are required. The primary key is made up of both these columns.

Tables are related by foreign keys

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

Example

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

- **dept_id** An ID number for the department. This is the primary key for the table.
- **dept_name** A column holding the name of the department.
- **dept_head_id** The employee ID for the department manager.

To find the name of a particular employee's department, there is no need to put the name of the employee's department into the employee table. Instead, the employee table contains a column holding the department ID of the employee's department. This is called a **foreign key** to the department table. A foreign key references a particular row in the table containing the corresponding primary key.

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

Other database objects

A relational database holds more than a set of related tables. Among the other objects that make up a relational database are:

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

  Indexes are an important design element for high performance, however their use is transparent to the user.

- **Views** Views are computed tables, or virtual tables. They look like tables to client applications, but they do not hold data. Instead, whenever they are accessed, the information in them is computed from the underlying tables.

  The tables that actually hold the information are sometimes called **base tables** to distinguish them from views.

- **Stored procedures and triggers** These are routines held in the database itself that act on the information in the database.

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

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

- **Users and groups** Each user of a database has a user ID and password. You can set permissions for each user, so that confidential information is kept private and users are prevented from making unauthorized changes. Users can be assigned to groups, in order to make the administration of permissions easier.

- **Java objects** You can install Java classes into the database. Java classes provide a powerful way of building logic into your database, and a special class of user-defined data types for holding information.

Queries

The "Q" in "SQL" stands for **query**. You query (retrieve) data from a database with the SELECT statement. The basic query operations in a relational system are projection, restriction, and join. The SELECT statement implements all of them.

Projections and restrictions

A **projection** is a subset of the columns in a table. A **restriction** (also called **selection**) is a subset of the rows in a table, based on some conditions.

For example, the following SELECT statement retrieves the names and prices of all products that cost more than $15:

```
SELECT name, unit_price
FROM product
WHERE unit_price > 15
```

This query uses both a restriction (WHERE unit_price > 15) and a projection (SELECT name, unit_price)

Joins

A join links the rows in two or more tables by comparing the values in key columns and returning rows that have matching values. For example, you might want to select the order item identification numbers and product names for all order items that were shipped more than a dozen items:

```
SELECT sales_order_items.id, product.name
FROM product KEY JOIN sales_order_items
WHERE sales_order_items.quantity > 12
```

The product table and the sales_order_items table are joined together based on the foreign key relationships between them.

Other SQL statements

You can do more with SQL than just query. SQL includes statements that create tables, views, and other database objects. It also includes statements that modify tables (the INSERT, UPDATE, and DELETE statements), and commands that perform many other database tasks discussed in this manual.

The system tables

Every database contains a set of **system tables**, these are special tables that the system uses to manage data and the system. These tables are also sometimes called the **data dictionary** or the **system catalog**.

System tables contain information about the database. You never alter the system tables directly in the way that you can alter other tables. The system tables hold information about the tables in a database, the users of a database, the columns in each table, and so on. This information is data about data, or **metadata**.