# Overview of SQL Server

In this chapter, I'll introduce you to Microsoft SQL Server 7, including the key features of the product. Then I'll cover its architecture and security model. Finally I'll cover specific issues related to using ADO to access SQL Server, such as building a connection string and mapping data types.

## Overview of SQL Server 7

Microsoft SQL Server 7 is Microsoft's premier database management system. It is easy to install and administer, and it comes with tools and wizards that make it easy to develop applications. As you might expect, SQL Server runs only on Microsoft Windows operating systems. Unlike previous versions of SQL Server, however, SQL Server 7 runs on both Windows 2000/NT and Windows 98/95 platforms.

SQL Server 7 has these key features:

- High performance relational database
- Scalable from small databases to very large databases
- Easy to install and use
- Reasonably priced
- Tightly integrated with Windows
- Built-in support for data warehousing

### SQL Server editions

SQL Server comes in three different editions: Desktop, Standard, and Enterprise. All three editions are built on the same code base, so applications that are developed to run on the Desktop Edition are guaranteed to be 100% compatible with the Standard and Enterprise Editions. Likewise, you may take an application running on the Enterprise Edition and run it on the Desktop Edition so long as you don't use any of the advanced features present in the Enterprise Edition.

## The Desktop Edition

The Desktop Edition is new in SQL Server 7 and is targeted at small databases that reside on a workstation. This edition might be useful for programmers who want to debug and test their applications locally before running them on a larger, shared database. You might also use it when you keep a local database to cache information or for use with a stand-alone application. You could also use database replication to keep this database synchronized with a larger database.

You can install the Desktop Edition only on Windows 98/95 systems, Windows NT Workstation and Windows 2000 Professional Systems. You are limited to a maximum of two CPUs in the same system and a maximum database size of 4GB. Also, some features that are found in the Standard Edition and Enterprise Edition, such as the OLAP Services and full-text indexing, aren't included in the Desktop Edition.

While the Desktop Edition uses the same code base as the Standard and Enterprise Editions, a lot of effort has gone into optimizing the database server for a workstation environment. The amount of main memory needed to run the database server has been minimized, and other changes have been made to get the optimal performance out of this configuration.

Caution

**Don't blame me:** While the Desktop Edition of SQL Server will run on a Windows 98/95 platform, you should not expect the same level of stability that you would find on a Windows 2000/NT platform.

## The Standard Edition

The Standard Edition is the traditional version of SQL Server that has been in use for years. This edition can be installed only on Windows 2000/NT Server. It can be used on systems with up to four CPUs, and there is no limit to the size of your database.

A lot of effort has gone into making SQL Server 7 easy to use. Many configuration options that existed in prior releases have been replaced with internal code that makes changes dynamically based on the workload. For parameters that still exist, intelligent wizards are available to help you choose the appropriate values.

In addition to all of the features present in the Desktop Edition, the Standard Edition includes support for parallel queries, read-ahead scans, and hash and merge joins, which makes your database programs much more efficient. Other features, such as full-text indexes, which make it easy to locate information in your database, and OLAP Services, which makes it easy to build data warehouses, are also included in the Standard Edition.

Note

**Data warehouses without programming:** Believe it or not, you can combine SQL Server Standard Edition with tools like Microsoft Excel 2000 and Microsoft MapPoint 2000 to create data warehouses without writing a single line of code. Don't believe me? Check out a copy of my book *Unlocking OLAP with SQL Server and Excel 2000* published by IDG Books Worldwide, Inc. I hope you like it.

### The Enterprise Edition

The Enterprise Edition is not for everyone. It is targeted at very large and/or high-activity database servers. It requires Windows 2000/NT Server, Enterprise Edition, which supports more than 2GB of main memory and more than four CPUs in a single server. It includes advanced features, such as fail over clustering facilities and the ability to partition an OLAP cube across multiple servers, which are designed to handle large workloads in a multiserver environment.

## SQL Server utilities

SQL Server includes several utilities to help you configure, manage, and use your database server. Of these tools, you're probably going to use the Enterprise Manager and the Query Analyzer most of all, with the Data Transformation Services (DTS) close behind. These are the tools with which you can create your database and its structures and extract information on the fly.

### Enterprise Manager

Enterprise Manager is a multipurpose utility that you use to manage your database. With this utility, you can start and stop the database server and set configuration properties for the database server. This utility allows you to define logon ids and manage security. You can even use it to create databases and their objects, including tables, indexes, stored procedures, and so on, using an interactive graphical design tool.

The Enterprise Manager uses the Microsoft Management Console (MMC) as its basic interface (see Figure 23-1). The display consists of an icon tree on the left side of the window and a display area that contains more detailed information about the selected icon in the icon tree.

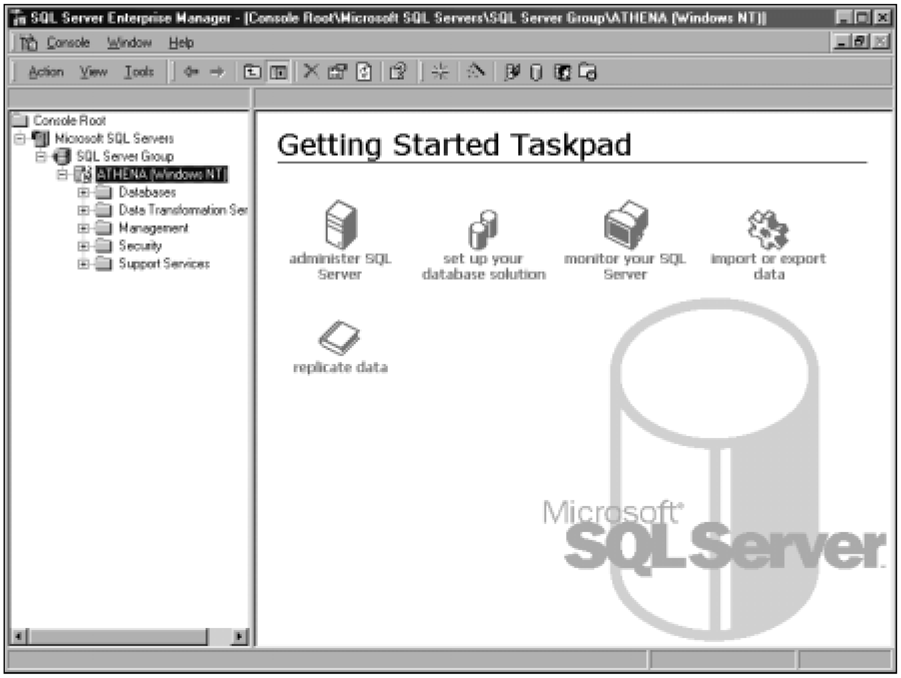
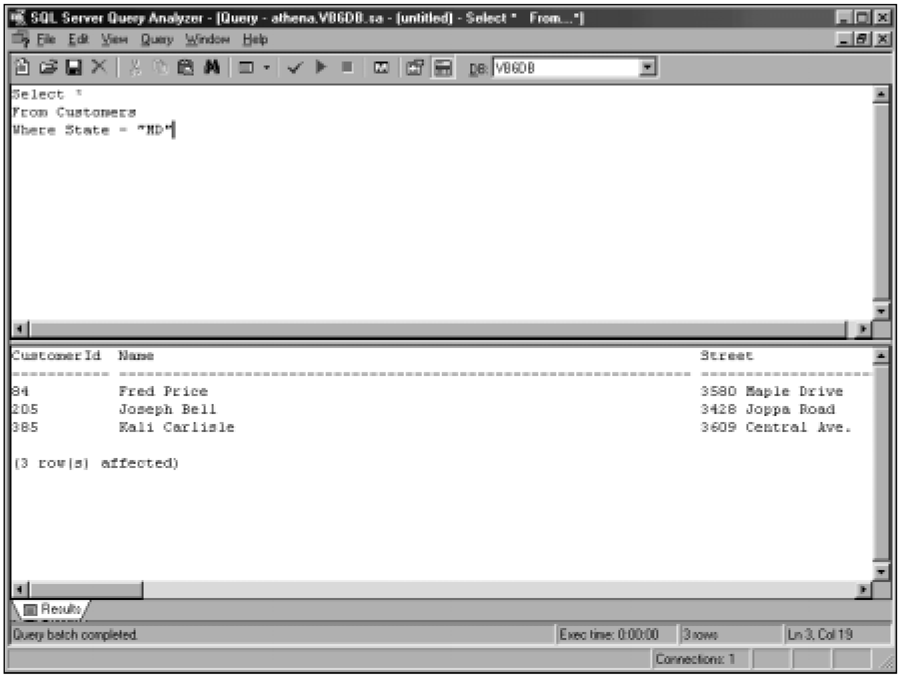


**Figure 23-1:** Running the Enterprise Manager

The Enterprise Manager contains a large number of wizards to help you perform common tasks such as creating databases, tables, and logon ids. Also, it contains wizards that help you define a maintenance schedule for tasks such as backing up your database, reorganizing indexes, and checking database integrity. Once the list of tasks has been selected, you then define a schedule for them, and SQL Server will take care of scheduling and running them. It will also keep records of the activities so that you can review them for potential problems.

## Query Analyzer

The Query Analyzer is a utility that allows you to execute SQL statements interactively, which is useful for testing SQL statements before you include them in your application (see Figure 23-2). You can also use this utility to extract information from the database, and you can use that information to validate the changes your application makes to your database.


SQL Server Query Analyzer - [Query - athena.VB6DB.sa - (untitled) - Select * From...*]
File Edit View Query Window Help
DB: VB6DB
Results

```
Select *
From Customers
Where State = "MD"
```

```
CustomerId  Name                                                           Street
----------- -------------------------------------------------------------- ------------------
84          Fred Price                                                     3580 Maple Drive
205         Joseph Bell                                                    3428 Joppa Road
385         Kali Carlisle                                                  3609 Central Ave.

(3 row(s) affected)
```

```
Query batch completed.    Exec time: 0:00:00    3 rows    Ln 3, Col 19
Connections: 1
```

**Figure 23-2:** Running the Query Analyzer

Tip

**Testing, testing, testing:** Since you can call stored procedures from within Query Analyzer, you may want to use it to test your stored procedures before trying to call them from your Visual Basic application.

## Data Transformation Services

I found that the Data Transformation Services is one of the most useful tools in SQL Server 7, since it makes it easy to move data from one place to another. It is extremely fast and flexible and can communicate with SQL Server databases, Oracle databases, Excel files, text files, and just about any other type of file that you can access using OLE DB or ODBC.

One of the nicest features included with DTS is the ability to transform your data while copying it. You can specify how each source column is mapped to the destination column. DTS will handle the most common data type transformations. DTS even has the ability to use a VBScript macro to transform one column to another, which may prove useful if a simple data type conversion isn't sufficient.

Tip

**But wait, there's more:** If your source and destination databases are both in SQL Server 7, you can copy all of the components of a database, including security information, stored procedures, and so on. This makes it easy to make a complete copy of a database for testing purposes.

### OLAP Services

OLAP (Online Analytical Processing) Services fulfills a critical need when implementing a data warehouse. It allows you to reorganize the data from a regular relational database into a multidimensional data store that can be easily used by data analysis tools such as Excel 2000. By including the ability to aggregate selected parts of the data, OLAP Services can offer much better performance than if a conventional database storage system were used.

### English Query

English Query is a series of COM components that allows you to translate an English language question into an SQL Statement. This ability makes it easy to build tools that let inexperienced users retrieve information from their databases. It also includes a tool that can automatically create a simple Web page that accepts questions and returns their results.

The English Query utility is used to collect information about the database you want to access. This utility is driven by a series of wizards that asks you detailed questions about the words that can be used to describe your data. This information is collected and saved in a domain that is used by the COM components to translate the user's question into a SQL **Select** statement.

## Database Architecture

SQL Server uses a true client/server architecture, where the database server runs in its own address space. While it can run on Windows 98/95 platform, you will get the best results running on multiprocessor computers with Windows 2000/NT Server. It is designed to be remotely administered and to exploit facilities that already exist in Windows such as e-mail, Internet access, etc.

SQL Server is arranged as a single Windows service that operates a collection of databases. Four of the databases are known as system databases, because they provide services critical to the database server itself. The other databases are known as user databases, because they contain user data.

### System databases

The system databases are *master*, *tempdb*, *msdb* and *model*. The master database contains information about the databases that the database server manages. This information includes the database name and physical files that are used. It also includes security information that allows users to access the database server itself.

Tempdb holds temporary tables and temporary stored procedures. It also supplies temporary storage for tasks such as holding the intermediate results of a query. Since none of the information stored in tempdb needs to exist beyond the current database server session, each time SQL Server is started, tempdb is erased.

The msdb database is used to maintain information about scheduled activities in the database, such as database backups, DTS jobs, and so on.

The model database is used as a template whenever SQL Server creates a new database. As part of the process to create a database, the contents of the model database are copied to the new database. Then the rest of the space in the database is filled with empty database pages.

## Database objects

Within a database there are several different types of database objects that can be manipulated. The most fundamental object within a database is a table. Tables come in two flavors, *system tables* and *user tables*. A system table contains information about the database. For instance a system table called `sysobjects` contains information about the various objects in the database. The database server itself updates system tables, though if you wish, you can read the information from the system tables to determine information about the database itself. User tables on the other hand are created by a database user and contain the user's information.

Caution

**Database destroyer:** Updating a system table directly is one of the fastest ways to destroy a database's integrity. Any mistake you may make may result in corrupting your database beyond repair. If you happen to be updating a system table in the master database, you've corrupted the entire database server.

Another type of database object is the *index*. The index holds information that allows the database server to quickly locate a row or set of rows in a table. Besides tables and indexes, objects such as stored procedures, users, roles, and so on are also stored in the database. However, unlike tables and indexes, which are stored directly in the database files, these objects are stored in system tables.

## Database storage

Each database is composed of two types of files: *data files* (`.MDF`) and *log files* (`.LDF`). Data files hold information kept in the database, while log files keep a history of the changes made to the database. A minimal database consists of one data file and one log file. Additional files can be used as needed. Data files consist of a series of 8KB pages. There are six types of pages (see Table 23-1). Each page has a 96-byte header, which contains system information like the type of page, the amount of free space on the page, and information about whose data is on the page. Pages can't be shared and are assigned to a single table or index or are used to hold allocation information.

Table 23-1
**Page Types**

| ***Page Type*** | ***Description*** |
|---|---|
| Data | Contains one or more of rows of information, except for large columns. |
| Index | Contains index entries. |
| Text/Image | Contains values stored in **Text, nText,** and **Image** columns. |
| Global Allocation Map | Contains information about allocated extents. |
| Index Allocation Map | Contains information about the extents used by a table or index. |
| Page Free Space | Contains information about the free space available on pages. |

A *data page* is used to store information about one or more rows. A row can't exceed the maximum size of a page. Since the page header size is 96 bytes and the additional information used to manage row information takes 32 bytes, the largest row you can store is 8,060 bytes. Note that if a row contains large columns (Text, nText, or Image), these columns do not count towards the 8,060-byte limit. Large columns are stored in their own pages and will occupy as many pages as needed to store the entire column's information.

An *extent* is eight pages. This is the primary unit of allocation for a table or index. If the size of a table or index is less than eight pages, it is assigned to a mixed extent that shares the extent with other tables and indexes. Once a table or index grows larger than eight pages, it is moved to a uniform extent, which is not shared with other database objects.

Tip

**For better performance:** If possible, you should keep your database files on an NTFS (NT file system) formatted disk drive with 64KB disk extents.

Log files consist of a series of log records. Each log record contains information necessary to undo a change made to a row. This consists of before and after values for every changed column, plus additional information that allows the records to be grouped together in transactions and to identify when and how the changes were made. When combined with a database backup, you can use the information in a log file to recover all of the changes to a database up to a specific point in time or when the database server stopped working.

## Database capacities

In an ideal world, a database would be able to store as much information as you would like. However, in the real world, there are always limits. Table 23-2 displays a list of limitations in SQL Server 7.

Table 23-2
**Database Capacities**

| ***Item*** | ***Capacity*** |
|---|---|
| Bytes per index | 900 |
| Bytes per key | 900 |
| Bytes per row | 8,060 |
| Clustered indexes per table | 1 |
| Columns per index | 16 |
| Columns per key | 16 |
| Columns per base table | 1,024 |
| Columns per **Select** statement | 4,096 |
| Columns per **Insert** statement | 1,024 |
| Database size | 1,048,516TB |
| Databases per server | 32,767 |
| Files per database | 32,767 |
| File size, data file | 32TB |
| File size, log file | 4TB |
| Foreign key table references per table | 253 |
| Identifier length in characters | 128 |
| Nested stored procedure calls | 32 |
| Nested subqueries | 32 |
| Nonclustered indexes per table | 249 |
| Parameters per stored procedure | 1024 |
| Rows per table | limited by available storage |
| Tables per **Select** statement | 256 |

## SQL Server data types

SQL Server supports a wide variety of data types as shown in Table 23-3.

Table 23-3
**SQL Server Data Types**

| ***SQL Server Data Type*** | ***Visual Basic Data Type*** | ***Comments*** |
|---|---|---|
| `Binary` | `Byte Array` | Contains a fixed length binary string up to 8,000 bytes long. |
| `Bit` | `Boolean` | **Null** values can't be used with a `Bit` field. |
| `Char` | `String` | Since `Char` fields always have a fixed length, the length of the `String` value will be the same as the length of the field. |
| `Datetime` | `Date` | A date and time value that is more accurate than the `Date` data type. The value will be rounded as needed. |
| `Decimal` | `Currency` | A packed decimal number with up to 38 digits of accuracy. Same as `Numeric`. |
| `Float` | `Double` | A 64-bit floating point number. |
| `Image` | `Byte Array` | A variable-length binary field containing up to $2^{31}$ –1 bytes. |
| `Int` | `Long` | A 32-bit integer. |
| `Money` | `Currency` | An 8-byte scaled integer with four digits of accuracy. Values can range from –922,337,203,685,477.5808 to +922,337,203,685,477.5807. |
| `NChar` | `String` | Contains a fixed length Unicode string up with up to 4,000 characters. |
| `Ntext` | `String` | Contains large blocks of Unicode text, with up to $2^{30}$ –1 characters. |
| `NvarChar` | `String` | Contains a variable-length Unicode string with up to 4,000 characters. |

| ***SQL Server Data Type*** | ***Visual Basic Data Type*** | ***Comments*** |
|---|---|---|
| `Numeric` | `Currency` | A packed decimal number with up to 38 digits of accuracy. Same as `Decimal`. |
| `Real` | `Single` | A 32-bit floating point value. |
| `SmallDatetime` | `Date` | A date and time value ranging from 1 Jan 1900 to 6 Jun 2079 and accurate to the minute. |
| `Smallint` | `Integer` | A 16-bit integer. |
| `SmallMoney` | `Currency` | A 4-byte scaled integer with four decimal digits of accuracy. Values can range from –214,748.3648 to +214,748.3647. |
| `Sysname` | `String` | `Sysname` is really a synonym for `Nchar(128)` and is used to hold the name of a database object. |
| `Text` | `String` | Contains large blocks of text data, up to $2^{31}$ –1 characters. |
| `Timestamp` | `Byte Array` | Contains a unique identifier that can be used to order a sequence of events. However, it doesn't contain a value that corresponds to a date or time. |
| `Tinyint` | `Byte` | An 8-bit integer. |
| `VarBinary` | `Byte Array` | If you declare a `Byte Array` without bounds and assign the value to it, you may then use the `Len` or `UBound` functions to determine the size of the field. |
| `VarChar` | `String` | The length of the `String` will be the length of the field. It has a maximum length of 8,000 characters. |
| `UniqueIdentifier` | `String` | Contains the hex equivalent of a GUID. |

## Connecting to SQL Server with ADO

Creating a connection string for SQL Server is a fairly straightforward process. You must specify the name of the provider, which is `SQLOLEDB` when connecting to an SQL Server database; the name of the data source, which is the name of the database server; and the initial catalog, which is the name of the name of the database you want to access. The first keyword in the list must be the `Provider=` keyword. The rest of the connection string is passed to the provider for interpretation. A sample connection string is shown below.

```
Provider=SQLOLEDB;Data source=Athena;Initial catalog=VB6DB
```

Besides specifying the database server and initial catalog, you can also include security information. If you're using SQL Server Authentication, you can include the `User Id=` keyword and the `Password=` keyword of the login you want to use to access the database. Otherwise, Windows NT Authentication will be used. You can also force the `SQLOLEDB` provided to use Windows NT Authentication by including `Trusted_Connection=yes` as part of the connection string.

Cross-Reference

Authentication is discussed in the next section, "SQL Server Security."

Tip

**Hacking security systems:** You should never hardcode a user id and password in a connection string. A hacker might view your program with a binary editor, which can display the contents of the program file in both ASCII and hex. A simple search on `SQLOLEDB` or `User Id` will let the hacker find these values. In the same vein, storing the user id and password in the Registry is also vulnerable to a smart hacker. Better alternatives would be to use Windows NT Authentication, which doesn't use user id and password to access the database, or to prompt the user for the user id and password and then insert them into the connection string.

## SQL Server Security

Everyone that accesses a SQL Server database must present login information that is verified before the user is granted access to the database server. This process is known as *authentication*.

### Authentication in SQL Server

SQL Server has two different modes of authenticating a user: SQL Server Authentication and Windows NT Authentication. You can use Windows NT Authentication by itself or in combination with SQL Server Authentication, which is also known as Mixed Mode Authentication. This authentication value is set for the entire database server level and affects all databases. While changing from Windows NT Authentication to Mixed Mode Authentication isn't difficult, converting Mixed Mode Authentication to Windows NT Authentication can be very difficult due to the number of logins that may need to be converted to Windows NT accounts.

Note

**Windows 2000 or NT only:** In order to use Windows NT Authentication, SQL Server must be running on a Windows 2000/NT platform. You must use SQL Server Authentication if you are running your database server on Windows 98/95.

## SQL Server Authentication

When a user connects to an SQL Server database using SQL Server Authentication, that user needs to provide a login id and a password. This information is presented when the user attempts to connect to the database. It is validated against the information in the master database. If the login information is correct, the user is granted access to the database server. If the login information is incorrect, the connection to the database is terminated.

Note

**No Windows, no options:** If you want to connect from a non-Windows computer to an SQL Server database, you must use SQL Server Authentication.

A SQL Server login can range in size from 1 to 128 characters and may contain any combination of letters, numbers, and special characters other than a backslash (\). From a practical viewpoint, using some special characters such as spaces in the login may force you to use double quotes (") or square brackets ([]) around the login id when you try to reference the login id in a SQL Statement. Logins are also case-insensitive, unless you selected a case-sensitive sort order when you installed SQL Server.

## Windows NT Authentication

Windows NT Authentication is based on Window 2000/NT login techniques. The same user name that you use to sign onto Windows becomes your login id. A password isn't needed, since the user name was validated when you signed onto Windows. However, just because your user name has been validated, doesn't mean that you automatically have access to SQL Server. SQL Server maintains information in the master database that identifies the users who are permitted to access the database server.

The primary advantage to using Windows NT Authentication is that you can use NT security groups. You can create a security group and add it to SQL Server in place of the user name. Then anyone who is a member of the security group automatically has access to the SQL Server database server.

Tip

**Simplify your services:** You should always use Windows NT Authentication for services such as an IIS Application or a COM+ transaction. This eliminates the need to store a database password in the application, which allows you to change passwords for these user names on a periodic schedule without recompiling the programs.

### Mixed Mode Authentication

Mixed Mode Authentication is a combination of both SQL Server Authentication and Windows NT Authentication. If a login id is specified when you connect to the database server, SQL Server Authentication is used. If no login id is specified, Windows NT Authentication is used. In both cases, the login information must be correct, or the connection to the database server will be dropped.

Note

**You may not have a choice:** While Windows NT authentication is more secure, you must use Mixed Mode Authentication if you plan to let users from non-Windows based computers access your database.

## SQL Server authorization

Just because the user has been granted access to the database, doesn't mean that the user can access any information inside the database or perform any database tasks. The user must be *authorized* to access resources and perform tasks. Each login must be mapped onto a specific user id in each database. Without this mapping, the login will be denied access to the database.

Within each database, each user is granted access to the various objects inside, such as tables, indexes, and stored procedures. The type of access granted to an object is known as a *permission*. The permissions available depend on the type of database object. For instance, on a table you can permit someone to select rows, insert rows, update rows, delete rows, or access the table as part of a referential integrity constraint.

In addition to granting access to database objects, you can also grant access to the SQL Statements that allow you to create, alter, and destroy various database objects, such as tables, views, and stored procedures. The same facility also controls the ability to back up and restore databases and database log files.

## SQL Server roles

In order to simplify security administration, SQL Server allows you to define *roles* in a database. Roles are similar to security groups in Windows 2000/NT. They represent a collection of users who are granted similar permissions. A role can be used in place of a user id when granting permissions.

Roles come in three flavors, *fixed server, fixed database,* and *user defined.* The fixed server roles determine the functions a login can perform at the database server level (see Table 23-4). The fixed database roles define groups of standard capabilities available to a user within a single database (see Table 23-5).

Table 23-4
**Fixed Server Roles**

| ***Role*** | ***Description*** |
|---|---|
| Sysadmin | has permission to perform any activity in SQL Server. |
| Serveradmin | has permission to set server-wide configuration options and to shut down the database server. |
| Setupadmin | has permission to manage linked servers and startup procedures. |
| Securityadmin | has permission to create and destroy logins, to set **Create Database** permissions, and to read error logs. |
| Processadmin | has permission to manage processes running in the database server. |
| Dbcreator | has permission to create and alter databases. |
| Diskadmin | has permission to manage the disk files used by SQL Server. |

Table 23-5
**Fixed Database Roles**

| ***Role*** | ***Description*** |
|---|---|
| db_owner | has permission to perform any activity and access any data in the database. |
| db_accessadmin | has permission to create and destroy user ids. |
| db_securityadmin | has permission to manage all database permissions, object ownerships, roles, and role memberships. |
| db_ddladmin | has permission to create and destroy database objects, but doesn't have permission to manage permissions. |
| db_backupoperator | has permission to backup the database. |
| db_datareader | has permission to read data from any user table in the database. |
| db_datawriter | has permission to modify data in any user table in the database. |
| db_denydatareader | has permission to deny read access to any database object. |
| db_denydatawriter | has permission to deny modify access to any database object. |

You can create user defined roles as needed to simplify security management. You can assign security permissions to a role, and every user associated with that role automatically inherits those permissions. As with fixed database roles, user defined roles are restricted to a single database.

Also, a user may be associated with multiple roles in the same database. This means that when you design your database, you can define roles for each basic function that a user can perform in the database. You can define roles that permit users to read information from different groups of tables, plus other roles that allow users to perform different types of updates. For instance, you may define a `Clerk` role that permits someone to enter order information, but prevents him or her from updating price information, while defining a `Manager` role that permits someone to change price information.

### Thoughts on SQL Server

SQL Server is a modern relational database management system that is more than competitive with any other database management system in the market today. It is well integrated into the Windows environment, which makes it ideal for those organizations where one or two people are responsible for the full range of IT services from IT Manager and Database Administrator to Systems Analyst, Programmer, and Computer Operator. The wide range of automated wizards allows you to install, build, and operate database applications much easier than with most other database management systems, yet SQL Server's performance is among the highest available, meaning that it will scale well into even the largest environments.

## Summary

In this chapter you learned:

- about SQL Server and its various editions.
- that Enterprise Manager is used to create and maintain SQL Server databases.
- that Query Analyzer is used to perform ad-hoc queries against an SQL Server database.
- that Data Transformation Services is a high performance data import and export utility.
- about the architecture of SQL Server.
- about the datatypes available in SQL Server.
- how to connect to SQL Server with ADO.
- how security is implemented in SQL Server.

✦ ✦ ✦