1

Introducing SQL Server 2005

To help you become familiar with SQL Server 2005, this chapter focuses on the key ingredients of the software. This chapter also outlines differences between different editions of the software before diving into particulars on the topics of architecture, database objects, databases, database storage, and server security. The chapter concludes with a brief look at the historical evolution of SQL Server.

What Is SQL Server 2005?

As you most likely know, SQL Server 2005 is primarily thought of as a *Relational Database Management System* (*RDBMS*). It is certainly that, but it is also much more.

SQL Server 2005 can be more accurately described as an *Enterprise Data Platform*. It offers many new features, and even more enhanced or improved features from previous editions of the product. In addition to traditional RDBMS duty, SQL Server 2005 also provides rich reporting capabilities, powerful data analysis, and data mining, as well as features that support asynchronous data applications, data-driven event notification, and more.

This book is primarily focused on the administration of the Database Engine. However, as mentioned, SQL Server 2005 includes many more features than just the relational engine. In light of that, it is important to start with some point of common reference. This section introduces the features of SQL Server 2005. It is not meant to be all-inclusive, but it will provide the context for the remainder of the book.

Later chapters go into greater detail and delve into the technologies behind each feature and how they affect you, the database administrator. SQL Server 2005 is such an enormous product that no one book could possibly cover every feature in detail, so some features will only be covered briefly as an introduction, while the core administrative features will be described in greater detail.

Database Engine

The Database Engine is the primary component of SQL Server 2005. It is the Online Transaction Processing (OLTP) engine for SQL Server, and has been improved and enhanced tremendously in this version. The Database Engine is a high-performance component responsible for the efficient storage, retrieval, and manipulation of relational and Extensible Markup Language (XML) formatted data.

SQL Server 2005's Database Engine is highly optimized for transaction processing, but offers exceptional performance in complex data retrieval operations. The Database Engine is also responsible for the controlled access and modification of data through its security subsystem. SQL Server 2005's Database Engine has many major improvements to support scalability, availability, and advanced (and secure) programming objects:

- *Physical partitioning of tables and indexes* — Tables and indexes can now be physically partitioned across multiple file groups consisting of multiple physical files. This dramatically improves the performance of data retrieval operations and maintenance tasks that are executed against very large tables. (See Chapter 5 for more information.)
- *Data Definition Language (DDL) triggers* — DDL triggers can be used to execute commands and procedures when DDL type statements are executed. In the past, modifications to the database could go undetected until they caused an application to fail. With DDL triggers, a history of all actions can be easily recorded or even prevented. DDL triggers can be placed at the server or database level.
- *Enhanced variable-length data types* — A new `MAX` keyword has been added to `varchar`, `nvarchar`, and `varbinary` data types that allow the allocation of up to 2GB of space for large object variables. One of the chief advantages of this addition is the ability to use large value types in the declaration and use of variables.
- *XML data type* — The new XML data type enables the storage of well-formed and schema-validated XML data. It also brings rich support in the form of XML data type methods, along with enhancements to `OPENXML` and `FOR XML` T-SQL commands.
- *Multiple Active Result Sets (MARS)* — MARS allows for clients to maintain more than one data request per connection. For example, in the past, if a connection was opened in an application, only one data reader could be opened to retrieve data from the database. To open another data reader, the first one had to be closed. With MARS, this limitation is removed.
- *Structured error handling* — T-SQL now includes the ability to perform structured error handling in the form of `TRY` and `CATCH` commands that remove the necessity of repeated checks for errors in scripts, and the ability to elegantly handle any errors that do occur.
- *Common Table Expressions (CTE)* — Microsoft has extended the American National Standards Institute (ANSI) compliance of T-SQL by including the ability to use the CTE object. CTEs are extraordinarily useful in the creation of efficient queries that return hierarchical information without the need for using lengthy and complicated recursive sub-queries.
- *Security enhancements* — SQL Server's security architecture has been enhanced considerably with the ability to enforce account policies on SQL Server logins. Other additions to SQL Server's security architecture include the control of execution context and the ability to create encryption keys and certificates to control access and guarantee the integrity of database objects through the use of digital signatures. See Chapter 6 for more information.

- *Common Language Run-Time (CLR) integration* — One of the most exciting additions to SQL Server is the integration of the CLR. It is also possibly the most misunderstood. The CLR provides a hosted environment for managed code. No longer is it necessary to make calls to external Application Programming Interfaces (API) via hard-to-manage extended stored procedures written and compiled utilizing unmanaged code to perform advanced and programmatic functions. Because the CLR is integrated in the Database Engine, database developers can now create secure and reliable stored procedures, functions, triggers, aggregates, and data types utilizing advanced C# and/or VB.NET features in the .NET Framework. The CLR in no way makes T-SQL obsolete, because T-SQL still out-performs managed code in the traditional manipulation of relational data. Where the CLR shines is in instances that require complex mathematical functions or that involve complex string logic. For an introductory look at the CLR see Chapter 12.

For complete coverage of the CLR check out the book, Professional SQL Server 2005 CLR Stored Procedures, Functions, and Triggers *by Derek Comingore (due for release by Wrox Press in the Fall of 2006).*

Analysis Services

Analysis Services delivers Online Analytical Processing (OLAP) and Data Mining functionality for business intelligence applications. As its name suggests, Analysis Services provides a very robust environment for the detailed analysis of data. It does this through user-created, multidimensional data structures that contain de-normalized and aggregated data from diverse data sources (such as relational databases, spreadsheets, flat files, and even other multidimensional sources).

The Data Mining component of Analysis Services allows the analysis of large quantities of data. This data can be "mined" for hidden relationships and patterns that may be of interest to an organization's data analyst. An example of this could be the online book store that analyzes your searches and purchases, comparing them to previous customers' search and purchase patterns to offer you suggestions or targeted advertisements. It could also be the cancer research group comparing health records and demographic data of patients to find some common pattern to the emergence of a particular form of cancer.

For a very detailed look at SQL Server 2005 Analysis Servers, check out the book, Professional SQL Server Analysis Services 2005 with MDX, *by Sivakumar Harinath and Stephen R. Quinn (Indianapolis: Wrox Press, 2006).*

Reporting Services

Reporting Services is a Web service–based solution for designing, deploying, and managing flexible, dynamic Web-based reports, as well as traditional paper reports. These reports can contain information from virtually any data source. Because Reporting Services is implemented as a Web service, it must be installed on a server with Internet Information Services (IIS). However, IIS does not have to be installed on a SQL Server. The Reporting Services databases are hosted on SQL Server 2005, but the Web service itself can be configured on a separate server.

For a detailed description of SQL Server 2005 Reporting Services and information about how to implement and extend SQL Server 2005 reports, check out an excellent book written by four very talented developers and personal friends, Professional SQL Server 2005 Reporting Services *(Indianapolis: Wrox Press, 2006). Paul Turley, Todd Bryant, James Counihan, and Dave DuVarney are amazing guys who I have had the great pleasure of working with over the past few years. You will not be disappointed.*

Integration Services

SQL Server Integration Services (SSIS) is Microsoft's new enterprise class data Extract, Transform, and Load (ETL) tool. SSIS is a completely new product built from the ashes of SQL Server 2000's Data Transformation Services (DTS). SSIS offers a much richer feature set and the ability to create much more powerful and flexible data transformations than its predecessor. This huge improvement, however, is not without a cost. SSIS is a fairly complex tool and offers a completely different design paradigm than DTS. Database administrators adept at the former tool are very often intimidated and frustrated by the new SSIS. Their biggest mistake is in thinking that Integration Services would just be an upgrade of Data Transformation Services. As stated previously, this simply isn't the case. More research, preparation, and training will be crucial to effectively utilizing SSIS. For an introductory look at SSIS, see Chapter 13.

For a very thorough discussion of this new feature of SQL Server 2005, read the excellent book, Professional SQL Server 2005 Integration Services *(Indianapolis: Wiley, 2006).*

Notification Services

Notification Services is used to build and deploy applications that support the generation and sending of data-driven notifications. Notification Services' applications provide the mechanism for subscribers to create a subscription for a specific event, which could be a database, file system, or some other programmatic event. The notification can take the form of an email or other custom delivery methods. For more information on Notification Services, see Chapter 14.

Service Broker

Service Broker provides the framework and services to enable the creation of asynchronous, loosely coupled applications. Service Broker implements a Service Orientated Architecture (SOA) in the data tier. It provides more controlled transaction-based communications than traditionally available in other SOA implementations such as Microsoft Message Queuing (MSMQ). Service Broker allows the developer to create database applications that focus on a particular task and allows the asynchronous communication with other applications that perform related (yet disconnected) tasks. For more information, see Chapter 15.

Data Tier Web Services

SQL Server 2005 provides support for creating and publishing data tier objects via HTTP without the use of an Internet Information Services (IIS) server. SQL Server 2005 can listen and respond to an HTTP port allowing developers to create applications that interact with a database across the Internet or through a firewall by using a Web service. For more information, see Chapter 7.

Replication Services

SQL Server 2005 Replication Services provides the ability to automate and schedule the copying and distribution of data and database objects from one database or server to another, while ensuring data integrity and consistency. Replication has been enhanced in SQL Server 2005 to include true Peer-to-Peer replication, replication over HTTP, the ability to replicate schema changes, and, very interestingly, the ability to configure an Oracle server as a replication publisher.

Multiple Instances

SQL Server 2005 provides the capability of installing multiple instances of the database application on a single computer. Depending on the edition of SQL Server being installed, up to 50 instances can be installed. This feature allows for one high-performance server to host multiple instances of the SQL Server services, each with its own configuration and databases. Each instance can be managed and controlled separately with no dependency on each other.

Database Mail

In the past SQL Server relied on a Messaging Application Programming Interface (MAPI) mail client configured on the server to facilitate email and pager notification for administrative and programmatic purposes. What this essentially meant was that to fully utilize administrative notifications, the administrator needed to install Outlook or some other MAPI-compliant client on the server, and then create a mail profile for the service account to use.

Many organizations wanted to take advantage of the SQL Server Agent's ability to send job and event notification via email but were unwilling to install unnecessary and potentially risky software on production server assets. The SQL Server 2005 Database Mail feature removes this requirement by supporting Simple Mail Transfer Protocol (SMTP) for all mail traffic. In addition, multiple mail profiles can be created in the database to support different database applications. For more information about Database Mail, see Chapter 8.

SQL Server 2005 Editions

SQL Server 2005 comes in six different flavors, and each has its specific place in the data management infrastructure with the probable exception of the Enterprise Evaluation Edition, which is only useful for short-term evaluation of the product (180 days). At the top of the list is the Enterprise Edition that supports absolutely everything that SQL Server 2005 has to offer. On the other end of the spectrum is the Express Edition, which offers very limited (but still exciting) features.

The following table contrasts the major differences between all but the Developer and Evaluation Editions. As discussed later in this section, the Developer Edition supports the same functionality as the Enterprise Edition, and the Evaluation Edition is the Enterprise Edition with a time-limited and restricted license.

Feature	Enterprise Edition	Standard Edition	Workgroup Edition
Failover Clustering	Yes	2-node	No
Multi-Instance Support	50	16	16
Database Mirroring	Yes	Limited	No
Enhanced Availability Features	Yes	No	No
Table and Index Physical Partitioning	Yes	No	No

Table continued on following page

Feature	Enterprise Edition	Standard Edition	Workgroup Edition
Analysis Services Support	Yes	Yes	No
Data Mining	Yes	Limited	No
Reporting Services	Yes	Limited	Very Limited
Notification Services	Yes	Limited	No
Integration Services	Yes	Limited	Very Limited
Replication Services	Yes	Limited	Limited

For a complete list of supported features consult SQL Server 2005 Books Online under the topic "Features Supported by the Editions of SQL Server 2005."

SQL Server 2005 Mobile Edition

SQL Server Mobile is the replacement for SQL Server CE first offered in SQL Server 2000. The Mobile Edition enables the installation of a small SQL Server database on a mobile device to support a CE or Windows mobile application. SQL Server Mobile also enables the support of a database that is replicated from a database hosted on a Windows Server.

This ability creates a world of opportunity for collecting data in a remote scenario and synchronizing that data with a land-based database. For example, consider an overnight delivery service that must maintain a record of a delivery truck's inventory, including packages delivered and picked up. The truck inventory could be uploaded via replication to a mobile device, where a mobile application kept track of the deliveries and new packages picked up at delivery locations. Once the truck came back to the delivery center, the mobile device could be synchronized with the central database via replication or data upload.

SQL Server 2005 Express Edition

SQL Express is at the lowest end of functionality and scalability, but I am very excited about this particular edition. SQL Express replaces the Microsoft Desktop Edition (MSDE) and has a similar price index — it's free. For its very low price (you can't beat free), it still contains a great deal of functionality.

The reason this edition excites me is that it is perfect for many of my customers who are starting or running small businesses. They have a genuine need for a centralized managed database, but aren't ready to pay for a more scalable and robust solution. At the risk of offending my friends in the Open Source community, most of my customers are not very technically savvy, and so very flexible and viable solutions like MySQL running on Linux or Windows is just not appropriate when a Database Engine with an intuitive and free graphical management tool exists.

One of the most exciting improvements to Microsoft's free version of its database system is that it comes with a graphical management environment. It also supports databases up to 4GB in size and contains much of the same functionality as the other editions.

SQL Express is a big step up from MSDE, its predecessor and is a very viable solution for standalone applications that require a managed data-store or even distributed applications with a minimal number of connections.

SQL Express can be installed on any Microsoft desktop or server operating system from Windows 2000 and beyond, so a very small company can still leverage the database technology without making a large investment. Once the company starts to grow, it will inevitably need to make the move to one of the more robust editions, but the upgrade process from SQL Express to its bigger siblings is a piece of cake because the data structures are nearly identical.

SQL Server 2005 Workgroup Edition

The Workgroup Edition replaces the SQL Server Personal Edition. It contains all the functionality of SQL Server 2005 Express Edition and then some. This edition is targeted to those small companies that have either outgrown the Express Edition or needed a more flexible solution to begin with, and yet do not need all the features of the Standard or Enterprise Edition.

The Workgroup Edition is very flexible and contains many of the features of the more expensive editions. What the Workgroup Edition doesn't provide is support for more advanced business intelligence applications, because SQL Server Integration Services and Analysis Services are not included in this edition. The Workgroup Edition also has a reduced feature set in regard to Reporting Services, but the Reporting Services features supported should satisfy most small organizations.

Like the Express Edition, the Workgroup Edition can be installed on both desktop and server operating systems, with the exception of Windows XP Home (which is not supported).

SQL Server 2005 Standard Edition

Most of the capabilities of SQL Server 2005 are supported in the Standard Edition, which makes it the ideal data platform for many organizations. What the Standard Edition does not provide are many of the features designed for the support of large enterprise databases. These features include many of the high-availability and scalability enhancements, such as Partitioned Tables and Parallel index operations. It also lacks some of the more advanced business intelligence features and Integration Services.

SQL Server 2005 Enterprise Edition

The Enterprise Edition is the full-meal deal. Nothing is held back. Parallel operations, physical table partitioning, complete business intelligence, and data mining support — you name it, the Enterprise Edition has it.

If you require an easy-to-implement-and-maintain platform that can support millions of transactions a second, 64 terabytes (TB) of RAM, and 64-bit processors, this release is for you. It is also an appropriate solution if you just require advanced business analytics, and not necessarily the millions of transactions a second that this edition offers.

Enterprise Edition is performance. Although the feature set between the Enterprise Edition and the Standard Edition is not huge, the differences in performance between the two editions can be. The Enterprise Edition fully optimizes read-ahead execution and table scans, which results in marked improvement in read and scan performance.

SQL Server 2005 Architecture

It is the job of SQL Server to efficiently store and manage related data in a transaction-intensive environment. The actual theories and principles of a relational database are beyond the scope of this book, and hopefully you already have some of that knowledge. What is pertinent to this book is the way SQL Server manages the data, and how it communicates with clients to expose the data. The following discussion describes the communication architecture utilized by SQL Server 2005, the services SQL Server 2005 utilizes, and the types of databases SQL Server uses. This section also discusses how those databases are stored and accessed, but you can find a detailed description of SQL Server 2005 storage architecture in Chapter 4.

SQL Server 2005 Communication

To adequately plan for a SQL Server database application, it is important to understand how SQL Server 2005 communicates with clients. As mentioned previously, SQL Server 2005 is more of a data platform than just a relational database server. Because the SQL Server 2005 platform offers several different data services, it also must provide different ways of accessing that data.

SQL Server 2005 ships with the ability to communicate over different protocols. By default, SQL Server will accept network connections via TCP/IP. The local Shared Memory protocol is also enabled by default to enable local connections without having to incur the overhead of a network protocol.

In addition to the TCP/IP, Named Pipes, and Shared Memory protocols, the Virtual Interface Adapter (VIA) protocol is available for VIA Storage Area Network (SAN) implementations.

With the exception of HTTP endpoints (described in Chapter 7), SQL Server utilizes a communication format called *Tabular Data Stream* (TDS). The TDS packets utilized by SQL Server are encapsulated in the appropriate protocol packets for network communication.

The task of wrapping the TDS packets is the responsibility of the SQL Server Network Interface (SNI) protocol layer. The SNI replaces the Server Net-Libraries and the Microsoft Data Access Components (MDAC) that were utilized in SQL Server 2000. SQL Server creates separate TDS endpoints for each network protocol.

Although TDS is the primary method for connecting to and manipulating data on a SQL Server, it is not the only method available. In addition to TDS communication, SQL Server 2005 supports native Data Tier Web services (see Chapter 7). By utilizing SQL Server Web services, connections can be made to SQL Server via any client application that supports HTTP and Simple Object Access Protocol (SOAP).

Supported Languages

SQL Server 2005 supports the following five different languages to enable data manipulation, data retrieval, administrative functions and database configuration operations:

- ❑ *Transact-Structured Query Language (T-SQL)* — This is Microsoft's procedural language extension to the Structured Query Language (SQL) standard established by the American National Standards Institute (ANSI). T-SQL is entry-level compliant with the ANSI-99 standard. T-SQL is the primary and most common method for manipulating data. For more information about T-SQL, consult *Beginning Transact-SQL with SQL Server 2000 and 2005* (Indianapolis: Wiley, 2005).

- *Extensible Markup Language (XML)* — This is fully supported in SQL Server 2005, as well as language extensions to XML that enable the retrieval and modification of data by utilizing XQuery syntax or native XML methods.
- The *Multidimensional Expressions (MDX)* — This language is used to query against multidimensional objects in SQL Server 2005 Analysis Services.
- *Data Mining Expressions (DMX)* — This is a an extension of Transact-SQL that enables the creation of queries against a data mining model implemented in SQL Server 2005 Analysis Services.
- *Extensible Markup Language for Analysis (XMLA)* — This can be used to both discover metadata from an instance of SQL Server 2005 Analysis Services and to execute commands against an instance of SSAS. XMLA commands are generally limited to the creation or modification of SSAS objects. Actual retrieval of SSAS data is done with MDX queries.

SQL Server Programming Object Models

Most of the administrative activity that must be done on SQL Server 2005 can be done using the provided tools, but sometimes it may be necessary to build custom administrative tools, or to be able to programmatically build and manipulate database objects. Three new object models have been created to support this need:

- *SQL Management Objects (SMOs)* — SMOs enable developers to create custom applications to manage and configure SQL Server 2005, SQL Server 2000, or SQL Server 7.0 Database Engines. It is an extensive library that provides full support for virtually all aspects of the relational store. The SMO library makes it possible to automate administrative tasks that an administrator must perform through custom applications, or with command-line scripts using the SMO `scripter` class.
- *Replication Management Objects (RMOs)* — RMOs can be used along with SMOs to implement and automate all replication activity, or to build custom replication applications.
- *Analysis Management Objects (AMOs)* — AMOs, like SMOs and RMOs, represent a complete library of programming objects. AMOs enable the creation of custom applications or automation of Analysis Server management.

SQL Server 2005 Services

SQL Server runs as a service. In fact, it runs as several services if all the different features of the product are installed. It is important to know what service is responsible for what part of the application so that each service can be configured correctly, and so that unneeded services can be disabled to reduce the overhead on the server and reduce the surface area of SQL Server.

MSSQLServer (SQL Server)

The `MSSQLServer` service is the database engine. To connect and transact against a SQL Server 2005 database, the `MSSQLServer` service must be running. Most of the functionality and storage features of the database engine are controlled by this service.

The `MSSQLServer` service can be configured to run as the local system or as a domain user. If installed on Windows Server 2003, it can also be configured to run under the Network System account.

SQLServerAgent (SQL Server Agent)

This service is responsible for the execution of scheduled jobs such as scheduled backups, import/export jobs, and Integration Services packages. If any scheduled tasks require network or file system access, the `SQLServerAgent` service's credentials are typically used.

The `SQLServerAgent` service is dependent on the `MSSQLServer` service. During installation, the option is given to configure both services with the same credentials. Although this is by no means required, it is common practice. A frequent problem encountered by database administrators is that jobs that work perfectly when run manually fail when run by the agent. The reason for the failure is because the account that is used when testing the job manually is the logged-in administrator, but when the job is executed by the agent, the account the agent is running under does not have adequate permissions.

MSSQLServerADHelper (SQL Server Active Director Helper)

Very often, the `MSSQLServer` service and the `SQLServerAgent` service are configured to run with a domain account that has local administrative rights on the server SQL Server is installed on. Although this configuration offers a great deal of flexibility to what the two services can do locally, it doesn't give them any permission to Active Directory.

In order for the `MSSQLServer` service to register its respective instance of SQL Server, it must be either running as the local system account (which significantly reduces the flexibility of the service), or be a member of the domain admin group (which grants it way too much access, violating the principle of least privilege).

To enable SQL Server to register itself in the domain, but not limit its functionality, the `MSSQLServerADHelper` service was created. The `MSSQLServerADHelper` service runs under the local system account of the domain computer SQL Server is installed on, and is automatically granted the right to add and remove objects from Active Directory. The `MSSQLServerADHelper` service only runs when needed to access Active Directory and is started by the `MSSQLServer` service when required. Regardless of the number of installed instances there is only one `MSSQLServerADHelper` service per computer.

MSSQLServerOLAPService (SQL Server Analysis Services)

`MSSQLServerOLAPService` is the service that Analysis Services runs under. Analysis Services provides the services and functionality to support all of SQL Server 2005's OLAP needs, as well as the new data mining engine included with SQL Server 2005.

SQLBrowser (SQL Server Browser)

The `SQLBrowser` service is used by SQL Server for named instance name resolution and server name enumeration over TCP/IP and VIA networks.

The default instance of SQL Server is assigned the TCP port 1433 by default to support client communication. However, because more than one application cannot share a port assignment, any named instances are given a random port number when the service is started. This random port assignment makes it difficult for clients to connect to it, because the client applications don't know what port the server is listening on. To meet this need, the `SQLBrowser` service was created.

On startup, the SQLBrowser service queries the registry to discover all the names and port numbers of installed servers and reserves UDP port 1434. It then listens on UDP port 1434 for SQL Server Resolution Protocol (SSRP) requests and responds to the requests with the list of instances and their respective port assignments so that clients can connect without knowing the port number assignment. There are definite security considerations to this arrangement, so it is very important that no unauthenticated traffic on UDP port 1434 be allowed on the network, because the service will respond to any request on that port. This creates the potential of exposing more information about the server instances than some organizations find acceptable.

If the SQLBrowser service is disabled, it will be necessary to specify a static port number for all named instances of SQL Service and to configure all client applications that connect to those instances with the appropriate connection information. For a full list of what features are affected by disabling the SQLBrowser, consult SQL Server 2005 Books Online.

MSFTESQL (SQL Server Full-Text Search)

The Microsoft Full-Text Engine for SQL Server (MSFTESQL) is used to support full-text indexing and full-text queries against text data stored in the database. The text data can be of several different data types including char, nchar, varchar, nvarchar, text, and ntext. In addition, full-text indexes can be created on binary formatted text such as Microsoft Word documents.

The chief advantage of the MSFTESQL service and associated engine is that it allows much more flexible and powerful searches against text data than the Transact-SQL LIKE command, which is limited to exact match searches. The MSFTESQL engine can perform exact match, proximity, linguistic, and inflectional searches. It will also exponentially outperform comparative Transact-SQL LIKE searches against large (millions of rows) tables. For a more complete discussion on both the Transact-SQL LIKE command and Full-Text search see *Beginning Transact-SQL with SQL Server 2000 and 2005* (Indianapolis: Wiley, 2005).

MSDTSServer (SQL Server Integration Services)

The MSDTSServer service provides management and storage support for SSIS. Although this service is not required to create, store, and execute SSIS packages, it does allow for the monitoring of SSIS package execution and displaying of a hierarchical view of SSIS packages and folders that are stored in different physical locations.

ReportServer (SQL Server Reporting Services)

The ReportServer service is the process in which Reporting Services runs. The service is accessible as a Web service and provides for report rendering, creation, management, and deploying. For more information on Reporting Services, see *Professional SQL Server 2005 Reporting Services* (Indianapolis: Wiley, 2004).

SQLWriter (SQL Server VSS Writer)

The SQLWriter service allows for the volume backup of SQL Server data and log files while the SQL Server service is still running. It does this through the Volume Shadow Copy Service (VSS). SQL Server database backups are typically performed through SQL Server's backup program or through third-party applications that communicate with SQL Server's backup program.

Normal system backups of volumes containing SQL Server log or data files will normally fail, because as long as SQL Server is running, the files are open. The SQLWriter service overcomes this limitation by allowing you to perform the backups with the VSS service. It is still recommended, however, to perform regular backups through SQL Server's backup program.

MSDTC (Distributed Transaction Coordinator)

The MSDTC service is used to manage transactions that span more than one instance of SQL Server or an instance of SQL Server and another transaction-based system. It utilizes a protocol known as Two-Phased Commit (2PC) to ensure that all transactions that span systems are committed on all participating systems.

SQL Server 2005 Database Objects

SQL Server 2005 database objects are defined and exist within a defined scope and hierarchy. This hierarchy enables more control over security permissions and organization of objects by similar function. SQL Server 2005 objects are defined at the Server, Database, and Schema levels.

Server

The server scope encompasses all the objects that exist on the instance of SQL Server, regardless of their respective database or namespace. The database object resides within the server scope.

One of the more confusing terms when working with SQL Server 2005 is the term *server*. When you hear the term "server," you often think of that piece of hardware taking up space on a *server* rack in the *server* room. Where the confusion arises is that you can install multiple instances of SQL Server on a single *server* (huh?).

What would probably be clearer is to say that the capability exists to install multiple instances of the SQL Server 2005 Data Platform application on a single computer running a Windows operating system. Though this might be more descriptive, it doesn't make for very interesting marketing material.

What is left is the fact that, when it comes to SQL Server 2005 and you read "server," it is important to check the context to make sure that it means an instance of SQL Server 2005 or the physical computer that SQL Server is installed on.

When it comes to the server scope and SQL Server 2005 database objects, the term "server" actually refers to the SQL Server 2005 instance name. In the majority of the examples in this book, the instance name is AUGHTFIVE, which is also the name of the server used in the writing of this book. So, the instance name AUGHTFIVE is the default instance installed on the Windows Server 2003 named AUGHTFIVE.

Database

The database scope defines all the objects within a defined database catalog. Schemas exist in the database scope.

The ANSI synonym for "database" is "catalog." When connecting to an instance of SQL Server 2005, it is generally desired to specify an Initial Catalog, or Initial Database. An instance of SQL Server 2005 can contain many databases. A typical database application is constrained within one database that contains all the data objects required to provide the functionality the application requires. This is not always the case, but it is the most common.

Schema

Each database can contain one or more schemas. A schema is a namespace for database objects. All data objects in a SQL Server 2005 database reside in a specific schema.

SQL Server 2005 implements the ANSI schema object. A database schema is a defined namespace in which database objects exist. It is also a fully configurable security scope. In previous releases of SQL Server, the namespace was defined by the owner of an object. In SQL Server 2005, the ownership of an object is separated from an object's namespace. An individual user may be granted ownership of a schema, but the underlying objects belong to the schema. This adds greater flexibility and control to the management and securing of database objects. Permissions can be granted to a schema, and those permissions will be inherited by all the objects defined in the schema.

Object Names

Every object in a SQL Server 2005 database is identified by a four-part, fully qualified name. This fully qualified name takes the form of `server.database.schema.object`. However, when referring to objects, the fully qualified name can be abbreviated. By omitting the server name SQL Server will assume the instance the connection is currently connected to. Likewise, omitting the database name will cause SQL Server to assume the existing connection's database context.

Omitting the schema name will cause SQL Server to assume the namespace of the logged-in user. This is where some confusion can be created. Unless explicitly assigned, new users are assigned the default schema of `dbo`. (See Chapter 6 for user and login management information.) As a result, all references to database objects not explicitly qualified will be resolved to the `dbo` schema.

For example, the user Fred logs in to the server `AUGHTFIVE` and his database context is set to `AdventureWorks`. Because Fred was not assigned a user-defined schema, he exists in the default `dbo` schema. Fred wants to retrieve the contents of the `Contact` table, so he executes the following query:

```
SELECT * FROM Contact;
```

Fred's query will resolve to `AUGHTFIVE.AdventureWorks.dbo.Contact`. Unfortunately, that table does not exist. The fully qualified name for the contact table is `AUGHT5.AdventureWorks.Person.Contact`. In order for Fred's query to work, one of two things will have to happen. The query will have to be rewritten to reference the appropriate schema scope, like the following example:

```
SELECT * FROM Person.Contact
```

Or, Fred's default schema can be changed to the `Person` schema so that his query will be properly resolved with the following command:

```
USE AdventureWorks;
GO
ALTER USER Fred WITH DEFAULT_SCHEMA=Person;
GO
```

Now, take a look at a different scenario. The user Fred is created and assigned the default schema of `Production`. Fred wants to retrieve the contents of a table called `dbo.HourlyWage` so he executes the following:

```
SELECT * FROM HourlyWage
```

SQL Server first resolves this query as `AUGHTFIVE.AdventureWorks.Production.HourlyWage` because Fred's default schema is `Production` and he did not explicitly tell SQL Server what schema to work with. Because the `HourlyWage` table does not exist in the `Production` schema, the initial resolution fails, but SQL Server then falls back to the `dbo` schema and resolves the name as `AUGHTFIVE .AdventureWorks.dbo.HourlyWage`. The resolution succeeds and Fred is returned the data he wanted.

SQL Server will always search the assigned schema first, then the `dbo` schema if the initial resolution fails. Care must be taken when creating objects so that the proper namespace is referenced. It is completely possible to create a table with the same name in two different schemas (for example, a `dbo.HourlyWage` and a `HumanResources.HourlyWage`). When this happens and an application is created to expose the contents of the `HourlyWage` table, the possibilities for inconsistencies and confusion are endless. If the schema is not referenced in the applications query, some users will invariably get their results from the table in the `dbo` schema, whereas others will end up getting results from the `HumanResources` version of the table. As a best practice, all objects should be referenced by a two-part name to avoid this confusion.

SQL Server 2005 Databases

There are two types of databases in SQL Server: system databases and user databases. The *system databases* are used to store system-wide data and metadata. *User databases* are created by users who have the appropriate level of permissions to store application data.

System Databases

The system databases are comprised of `Master`, `Model`, `MSDB`, `TempDB`, and the hidden `Resource` database. If the server is configured to be a replication distributor, there will also be at least one system distribution database that is named during the replication configuration process.

The Master Database

The `Master` database is used to record all server-level objects in SQL Server 2005. This includes Server Logon accounts, Linked Server definitions, and EndPoints. The `Master` database also records information about all the other databases on the server (such as their file locations and names). Unlike its predecessors, SQL Server 2005 does not store system information in the `Master` database, but rather in the `Resource` database. However, system information is logically presented as the `SYS` schema in the `Master` database.

The Model Database

The `Model` database is a template database. Whenever a new database is created (including the system database `TempDB`), a copy of the `Model` database is created and renamed with the name of the database being created. The advantage of this behavior is that objects can be placed in the `Model` database prior to the creation of any new database and, when the database is created, the objects will appear in the new database. For example, it has always bugged me that Transact-SQL does not contain a `Trim` function to truncate both leading and trailing spaces from a string of characters. Transact-SQL offers an `RTRIM` function that truncates trailing spaces and an `LTRIM` function that removes leading spaces. The code to successfully implement a traditional trim operation thus becomes the following:

```
LTRIM(RTRIM('character string'))
```

To reduce my irritation level and the number of characters I needed to type to successfully trim a character string, I created my own `TRIM` function in the `Model` database with the following code:

```
USE Model
GO
CREATE FUNCTION dbo.Trim (@String varchar(MAX))
RETURNS varchar(MAX)
AS
BEGIN
  SELECT @String = LTRIM(RTRIM(@String))
  RETURN @String
END
```

After creating this function in the `Model` database, it will be propagated to all databases created after adding it to the `Model` database and can be utilized with the following simplified code:

```
dbo.TRIM('character string')
```

I know it's only a saving of two characters, but those two characters are open and close parenthesis characters, which are often the source of annoying syntax errors. By reducing the nested functions, the overall complexity of the function call is also reduced.

Almost any database object can be added to the `Model` database so that they are available in subsequently created databases. This includes database users, roles, tables, stored procedures, functions, and assemblies.

The MSDB Database

I mostly think of the `MSDB` database as the SQL Server Agent's database. That's because the SQL Server Agent uses the `MSDB` database extensively for the storage of automated job definitions, job schedules, operator definitions, and alert definitions. The SQL Server Agent is described in greater detail in Chapter 8, but for now, just know that the Agent is responsible for almost all automated and scheduled operations.

The SQL Server Agent is not the only service that makes extensive use of the `MSDB` database. Service Broker, Database Mail, and Reporting Services also use the `MSDB` database for the storage of scheduling information. In addition to automation and scheduling information, SQL Server Integration Services (SSIS) can also utilize the `MSDB` database for the storage of SSIS packages.

The TempDB Database

The `TempDB` database is used by SQL Server to store data — yes, you guessed it, temporarily. The `TempDB` database is used extensively during SQL Server operations, so careful planning and evaluation of its size and placement are critical to ensure efficient SQL Server database operations.

The TempDB database is used by the Database Engine to store temporary objects (such as temporary tables, views, cursors, and table-valued variables) that are explicitly created by database programmers. In addition, the TempDB database is used by the SQL Server database engine to store work tables containing intermediate results of a query prior to a sort operation or other data manipulation. For example, if you wrote a query that returned 100,000 rows and you wanted the results sorted by a date value in the results, SQL Server could send the unsorted results to a temporary work table where it would perform the sorting operation and then return the sorted results to you. The TempDB database is also used extensively to support new connection options such as SNAPSHOT ISOLATION or Multiple Active Result Sets (MARS). If online index operations are performed, the TempDB database will hold the index during the build or rebuild process.

Another important aspect to keep in mind about the TempDB database is that all database users have access to it and have the ability to create and populate temporary objects. This access can potentially create locking and size limitation issues on SQL Server, so it is important to monitor the TempDB database just like any other database on SQL Server.

The Resource Database

The last system database is the Resource database. The Resource database is a read-only database that contains all the system objects used by an instance of SQL Server. The Resource database is not accessible during normal database operations. It is logically presented as the SYS schema in every database. It contains no user data or metadata. Instead, it contains the structure and description of all system objects. This design enables the fast application of service packs by just replacing the existing Resource database with a new one. As an added bonus, to roll back a service pack installation, all you have to do is replace the new Resource database with the old one. This very elegant design replaces the older method of running many scripts that progressively dropped and added new system objects.

User Databases

User databases are simply that: databases created by users. They are created to store data used by data applications and are the primary purpose of having a database server. During installation, you have the option of installing two sample user databases: AdventureWorks and AdventureWorksDW.

The AdventureWorks database is an OLTP database used by the fictitious Adventure-Works Cycles Company, which sells mountain bikes and mountain-biking-related merchandise.

The AdventureWorksDW database is an OLAP database used for data analysis of historical Adventure-Works Cycles data. Most of the sample code and examples provided in Books Online use these two sample databases.

Distribution Databases

One or more distribution databases can be configured to support replication. Some SQL Server professionals describe the distribution databases as system databases, and yet others describe them as user databases. I don’t think it makes much difference. What is important is what the database or databases do.

The distribution database stores metadata and transactional history to support all types of replication on a SQL Server. Typically, one distribution database is created when configuring a SQL Server as a replication Distributor. However, if needed, multiple distribution databases can be configured.

A model distribution database is installed by default and is used in the creation of a distribution database used in replication. It is installed in the same location as the rest of the system databases and is named `distmdl.mdf`.

SQL Server 2005 Database Storage

All system and user databases (including the `Resource` database) are stored in files. There is always a minimum of two files: one data file and one transaction log file. The default extension for data files is `.mdf`, and the default for transaction log files is `.ldf`.

The default location for the system database files is `<drive>:\Program Files\Microsoft SQL Server\MSSQL.X\MSSQL\Data\`, where `<drive>` is the installation drive and `X` is the instance number (`MSSQL.1` for the first instance of the database engine). The following table lists the names and default locations for system database files associated with the first instance of SQL Server.

System Database	Physical Location
Master	<install path>\MSSQL.1\MSSQL\Data\master.mdf
	<install path>\MSSQL.1\MSSQL\Data\mastlog.ldf
Model	<install path>\MSSQL.1\MSSQL\Data\model.mdf
	<install path>\MSSQL.1\MSSQL\Data\modellog.ldf
MSDB	<install path>\MSSQL.1\MSSQL\Data\msdbdata.mdf
	<install path>\MSSQL.1\MSSQL\Data\msdblog.ldf
TempDB	<install path>\MSSQL.1\MSSQL\Data\tempdb.mdf
	<install path>\MSSQL.1\MSSQL\Data\templog.ldf
Resource	<install path>\MSSQL.1\MSSQL\Data\Mssqlsystemresource.mdf
	<install path>\MSSQL.1\MSSQL\Data\Mssqlsystemresource.ldf

When it comes to the system databases, the following guidance is given: *Don't mess with them*. Your ability to manipulate the system databases in SQL Server 2005 has been extremely limited by the developers at Microsoft. Overall, this is a good thing. Generally speaking, the only thing you are permitted to do with system databases is back them up or move them to faster, more reliable disk arrays if they prove to be a performance bottleneck. The ability to modify the data contained in system tables through ad hoc updates that existed in prior releases has been almost completely removed from SQL Server 2005. To modify the system catalog, the server must be started in Single-User mode and even then, activity is restricted and is not supported by Microsoft.

Data Files and Filegroups

When a user database is created, it must contain at least one data file. This first data file is known as the *primary data file*. The primary data file is a member of the default *Primary filegroup*. Every database has one Primary filegroup when created and is made up of at least the primary data file. Additional data files can also be added to the Primary filegroup. More filegroups can also be defined upon initial creation of the database, or added after the database is created. Chapter 4 describes the storage architecture of files in greater detail, and Chapter 5 explains the advantage of filegroups. For now, it is sufficient to know that all of the data objects in a database (such as tables, views, indexes, and stored procedures) are stored within the data files. Data files can be logically grouped to improve performance and allow for more flexible maintenance (see Figure 1-1).

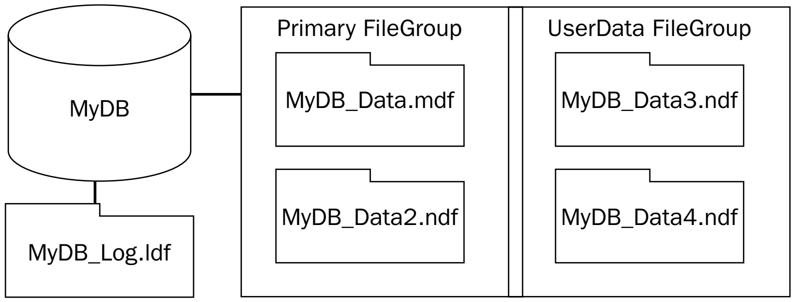

Figure 1-1: Data files and filegroups

Log Files

Upon initial creation of a database, one transaction log must be defined. The transaction log is used to record all modifications to the database to guarantee transactional consistency and recoverability.

Although it is often advantageous to create multiple data files and multiple filegroups, it is very rarely necessary to create more than one log file. This is because of how SQL Server accesses the files. Data files can be accessed in parallel, enabling SQL Server to read and write to multiple files and filegroups simultaneously. Log files, on the other hand, are not accessed in this manner. Log files are serialized to maintain transactional consistency. Each transaction is recorded serially in the log in the sequence it was executed. A second log file will not be accessed until the first log file is completely filled. You can find a complete description of the transaction log and how it is accessed in Chapter 4.

SQL Server Security

Chapter 6 provides a thorough discussion of SQL Server 2005 security features. However, to select the proper authentication model during installation, it is important to have a basic understanding of how SQL Server controls user access.

SQL Server 2005 can be configured to work in either the Windows Authentication Mode or the SQL Server and Windows Authentication Mode, which is also frequently called Mixed Mode.

Windows Authentication Mode

In Windows Authentication Mode only logins for valid Windows users are allowed to connect to SQL Server. In this authentication mode, SQL Server "trusts" the Windows, Windows Domain, or Active Directory security subsystem to have validated the account credentials. No SQL Server accounts are allowed to connect. They can be created, but they cannot be used for login access.

SQL Server and Windows Authentication Mode (Mixed Mode)

In SQL Server Mode and Windows Authentication Mode or Mixed Mode, valid Windows accounts and standard SQL Server logins are permitted to connect to the server. SQL Server logins are validated by supplying a username and password. Windows accounts are still trusted by SQL Server. The chief advantage of Mixed Mode is the ability of non-Windows accounts (such as UNIX) or Internet clients to connect to SQL Server.

A (Very) Brief History of SQL Server

How did we get here? Where did SQL Server 2005 come from? Without spending a great deal of time discussing the complete history of SQL Server, I thought it would be of some interest to give a very brief overview of SQL Server's roots. I often joke with colleagues and customers that some day I'm going to write a "Trivial Pursuit, Geek Edition." This short description may help you get the yellow history wedge, so pay close attention!

In the Beginning

Microsoft's foray into the enterprise database space came in 1987 when it formed a partnership with Sybase to market Sybase's DataServer product on the Microsoft/IBM OS/2 platform. From that partnership, SQL Server 1.0 emerged, which was essentially the UNIX version of Sybase's DataServer ported to OS2.

The Evolution of a Database

After a number of years, the developers at Microsoft were allowed more and more access to the Sybase source code for test and debugging purposes, but the core SQL Server application continued to be a product of Sybase until SQL Server 4.2 was released for Windows NT in March of 1992.

SQL Server 4.2 was the first true joint product developed by both Sybase and Microsoft. However, the database engine was still pure Sybase. Only the tools and database libraries were developed by Microsoft. Up to that point, SQL Server had been developed to run primarily on the OS/2 platform, but with the release of Windows NT, the developers at Microsoft essentially abandoned any OS/2 development and focused on bringing a version of SQL Server to Windows NT.

Microsoft Goes It Alone

With the growing success of Sybase in the UNIX market and Microsoft in Windows, the two companies found themselves competing for market share on a product essentially developed by Sybase. As a result, in 1994, the two companies terminated their joint development agreement, and Sybase granted Microsoft a limited license to use and modify Sybase technology exclusively for systems running on Windows.

A year later, in June 1995, Microsoft released the first version of SQL Server developed exclusively by Microsoft developers — SQL Server 6.0 — but the core technology was still largely Sybase code-base. Less than a year later, more changes were made and Microsoft released SQL Server 6.5 in April of 1996.

Meanwhile, the developers on the SQL Server team were beginning work on a new database system code-named "Sphinx." The Sybase code-base was rewritten almost from scratch for Sphinx, and only a handful of code remained to indicate SQL Server's humble beginnings in OS/2.

In December of 1998, Sphinx was officially released as SQL Server 7.0. The changes from SQL Server 6.5 were readily apparent from the first second a database administrator launched the new Enterprise Manager. Finally, there was a robust and reliable database system that was easy to manage, easy to learn, and still powerful enough for many businesses.

As SQL Server 7.0 was being released, the next version was already in development. It was code-named "Shiloh." Shiloh became SQL Server 2000 and was released in August of 2000. The changes to the underlying data engine were minimal, but many exciting changes that affected SQL Server's scalability issues were added (such as indexed views and federated database servers), along with improvements like cascading referential integrity. Microsoft's enterprise database server was finally a true contender in the marketplace.

Back at Microsoft, the SQL team was working on an even more powerful and exciting release code-named "Yukon," which is now SQL Server 2005. After more than five years in development, a product that some were calling "Yukon the giant (Oracle) killer" was finally released. It is indeed a very significant release, and only time will tell how successful Microsoft is with it.

So, now, without further delay, the remainder of this book will be dedicated to introducing you to this very exciting and capable database management system.

Summary

This chapter introduced the basic structure and purpose of SQL Server 2005, along with a brief explanation of the various features available in this release of Microsoft's database application. Subsequent chapters delve into the technologies and features exposed in this chapter so that the database administrator can better understand and implement each feature introduced.

In Chapter 2, you learn how to plan and perform a SQL Server 2005 installation. Included in the discussions are prerequisite hardware and software configurations, as well as service and security considerations. A thorough installation plan will always reap enormous benefits when it comes to post-installation modifications. Understanding what to install (and how to install it) is invaluable.