1

Introducing SQL Server 2008

Before getting into the meat (or tofu, if you prefer) and potatoes of SQL Server 2008, it's important that you understand what exactly it is that you have on your plate. In this chapter, you will learn about the history of SQL Server, the key components of SQL Server, and the different editions, or *flavors*, of SQL Server. This chapter also covers architecture, database objects, database storage, and server security from a very high level, with more detail to follow in subsequent chapters.

A Condensed History of SQL Server

Now that the world revolves around SQL Server (at least, it feels that way, doesn't it?), it's interesting to trace Microsoft SQL Server 2008 back to its humble origins. While this is by no means a comprehensive history of SQL, it does provide some insight into the evolution of the product, as well as an idea of where it might be headed. And who knows? This bit of trivia may still show up in *Trivial Pursuit: Geek Edition* for a yellow pie slice.

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 OS/2.

The Evolution of a Database

After several 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 1992.

SQL Server 4.2 was the first true joint product developed by both Sybase and Microsoft. The Database Engine was still Sybase, but the tools and database libraries were developed by

Microsoft. While SQL Server had been developed to run primarily on the OS/2 platform, the release of Windows NT heralded in a new era. 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 1996.

With SQL Server 6.5 complete, 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 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 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.

Over the next few years, 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.

BI for the Masses

While calling SQL Server 2005 an ''Oracle killer'' might have been a bit optimistic, no one can deny the broad appeal of SQL Server 2005 as a great leap forward for Microsoft. Since its release, it has been the core technology behind a great number of Microsoft products, including SharePoint, PerformancePoint, and the System Center family of products. Many third-party vendors have also leveraged SQL for ERP systems and other software products.

Where SQL Server 2005 really stood apart from its competitors was in its Business Intelligence (BI) offerings. These include tools for moving and transforming data (SQL Server Integration Services), analyzing data (SQL Server Analysis Services), and reporting on data (SQL Server Reporting Services). These three components, in addition to Notification Services and the Service Broker, were part of Microsoft's commitment to make SQL Server 2005 stand out as more than just a Database Engine. The inclusion of these technologies made SQL Server 2005 extremely attractive to businesses that were just starting to discover and utilize BI.

2008 . . . and Beyond!

In August 2008, Microsoft SQL Server 2008 was released to manufacturing (RTM). While SQL Server 2008 isn't as much of a paradigm shift from SQL Server 2005 as its predecessor was from SQL Server 2000, it contains many improvements and new features that make it a compelling upgrade (which we will cover throughout this book). SQL Server 2000 reached its end-of-life mainstream support in April 2008, which should also help drive the adoption of SQL Server 2008.

Microsoft has invested heavily in SQL Server as a core technology and key platform, and there doesn't appear to be any slowdown in the near future. Rumors continue to persist that Microsoft Exchange and Active Directory, as well as a new file system, will leverage SQL Server 2008's Database Engine.

What Is SQL Server 2008?

As you most likely know, SQL Server 2008 is primarily thought of as a *Relational Database Management System* (*RDBMS*). It is certainly that, but it is also much more.

SQL Server 2008 can be more accurately described as an *Enterprise Data Platform*. It builds on many of the features that had first been incorporated in SQL Server 2005, while also expanding its offerings to include several improvements and additions. Primarily known for its traditional RDBMS role, SQL Server 2008 also provides rich reporting capabilities, powerful data analysis, and data mining. It also has 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 2008 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 2008. It is not meant to be all-inclusive, but it will provide the context for the remainder of the book.

Database Engine

The Database Engine is the primary component of SQL Server 2008. It is the Online Transaction Processing (OLTP) engine for SQL Server and has received further enhancements since SQL Server 2005. 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 2008'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. The Relational Database Engine in SQL Server 2008 has many improvements to support scalability, availability, security, and programmability. The following list is by no means a comprehensive list, but just a short overview of what's new in SQL Server 2008:

❑ **Hot Add CPU** — If your hardware or software environment supports it, SQL Server 2008 will allow you to dynamically add one or more CPUs to a running system. These CPUs can be physical, logical, or virtual.

❑ **Option to Optimize for Ad Hoc Workloads** — SQL Server 2008 includes a new feature that allows administrators to configure the server to improve plan cache efficiency for ad hoc batches. With this feature enabled, the Database Engine no longer needs to store fully compiled plans that will not be reused. Instead, the plan cache stores a stub of the ad hoc workload.

Chapter 1: Introducing SQL Server 2008

❑ **SQL Server Extended Events** — SQL Server 2005 introduced the ability to associate SQL Profiler traces with Windows Performance Log counters. This was extremely helpful in identifying poorly performing queries or the lack of sufficient resources in the system to handle certain events. SQL Server 2008 takes this a step further by introducing SQL Server Extended Events. Extended events allow database administrators to get a better understanding of the system behavior by correlating SQL Server data to the operating system or database applications. This is handled by directing output from extended events to Event Tracing for Windows (ETW).

❑ **Resource Governor** — The Resource Governor is a new feature that allows administrators to specify configuration options that limit the amount of CPU and memory available to incoming requests. This can help prevent applications or queries from consuming 100 percent of the CPU or all available memory. The Resource Governor uses configurable workload groups, which define what the CPU and memory limits are for any session that is classified as being a member of that group. Classification can be performed based on a number of system functions or user-defined functions.

❑ **Policy-Based Management** — SQL Server 2008 includes features that allow administrators greater control over their server environments by enforcing behaviors or constraints through a policy-based mechanism. In addition to using the included policies, administrators can create their own policies to configure servers to meet compliance requirements and standardize naming conventions, thereby simplifying administration.

❑ **Centralized Management** — Central Management servers are SQL Servers that can be configured to manage multiple servers as part of a group. You can also execute queries against a SQL Server group that can return results to either a combined set or a separate pane per server. A Central Management server can also be used to enforce management policies against multiple target servers simultaneously.

❑ **Query Editor IntelliSense** — SQL Server Management Studio now provides IntelliSense functionality in the Query Editor. The IntelliSense functionality provides auto-completion ability, error underlining, quick info help, syntax pair matching, and parameter help.

❑ **PowerShell Provider** — SQL Server 2008 includes new features that integrate with Windows PowerShell to help administrators automate many SQL Server 2008 tasks. *PowerShell* is an administrative command-line shell and scripting language that can make it easier to perform many common tasks through automation. The PowerShell provider in SQL Server 2008 exposes SQL Server Management Objects (SMO) in a structure similar to file system paths. SQL Server PowerShell also includes several SQL Server cmdlets for running scripts and other common tasks.

❑ **Compressed Indexes and Tables** — Compression is now supported for tables, indexes, and indexed views on either rows or pages. Compression operations will have an effect on performance. Because of this, row and page compression can be configured on a per-partition basis. For example, you could choose to compress a Read Only partition, but leave a Write-intensive partition uncompressed to minimize impact on the CPU.

❑ **FILESTREAM** — FILESTREAM is a new storage mechanism for storing data on the file system, rather than in the database itself. SQL Server 2008 applications can use FILESTREAM to take advantage of the storage and performance benefits of the NTFS file system while maintaining transactional consistency with the files themselves. Developers can leverage FILESTREAM as a mechanism for allowing large files to be maintained by the application database, without causing the database to become unnecessarily bloated. (Although this is just speculation on my part, I would be surprised if future releases of SharePoint didn't leverage FILESTREAM storage.)

4

❑ **Partition Switching** — Simply put, Partition Switching enables you to move data between partitions for a table or index. Data can be transferred between partitions without disrupting the integrity of the table or index.

❑ **Spatial Data Types** — Two new data types have been created for storing planar, or ''flat-earth'' data as well as ellipsoidal, or ''round-earth'' data. These data types are known as the geometry data type and geography data type, respectively.

❑ MERGE **Statement** — Transact-SQL includes a new MERGE statement that, based on the results of a join with a source table, can perform INSERT, UPDATE, or DELETE operations against a target table. For example, you can use MERGE to incrementally update a destination table by comparing the differences from a source table.

Integration Services

SQL Server Integration Services (SSIS) is Microsoft's enterprise class data Extract, Transform, and Load (ETL) tool. SSIS was originally introduced in SQL Server 2005 as a significant re-design 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, and this has been further expanded in SQL Server 2008. As another component in SQL Server's BI stack, SSIS provides a rich environment for moving and transforming data from a variety of source and destinations systems. SSIS 2008 includes performance enhancements, new ADO.NET components, and a new script environment that integrates with Visual Studio Tools for Applications (VSTA). SSIS is covered in more detail in Chapter 16.

For a very thorough discussion of this feature of SQL Server 2008, read the book by Brian Knight, Erik Veerman, Grant Dickinson, Douglas Hinson, and Darren Herbold, Professional Microsoft SQL Server 2008 Integration Services (Wiley, 2008).

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 other multidimensional sources). Unlike the OLTP engine, which is optimized for Write performance, the OLAP engine is optimized for Read performance, allowing queries and reports to return results from millions of rows of data in a short period of time, with minimal (if any) impact to the OLTP engine.

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. In fact, a well-known story about data mining involves Wal-Mart, Pop-Tarts, and hurricanes. Data mining revealed that prior to a hurricane event, sales of Pop-Tarts, particularly strawberry Pop-Tarts, would surge. In fact, they concluded that sales of strawberry Pop-Tarts before a hurricane were seven times that of normal sales. This allowed Wal-Mart to plan their inventory accordingly and meet customer demands. What's interesting about this example is that it seems completely random. Data mining does not attempt to answer the question of why Strawberry Pop-Tarts sell at a much higher rate, and sometimes, it may not even matter. Data mining revealed an absolute and consistent pattern that was used to plan inventory accordingly and paid off for the retailer.

Chapter 1: Introducing SQL Server 2008

A more detailed introduction to SSAS and Data Mining is in Chapter 17.

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. Although Reporting Services is implemented as a Web Service, it does not depend on Internet Information Services (IIS). In fact, in SQL Server 2008, IIS is no longer used to manage SQL Server Reporting Services. SQL Server Reporting Services (SSRS) can publish reports to a Virtual Directory hosted by the SQL Server itself, or to a SharePoint library. More information about SSRS can be found in Chapter 18.

> *For a detailed description of SQL Server 2008 Reporting Services and information about how to implement and extend SQL Server 2008 reports, check out an excellent book,* Professional Microsoft SQL Server 2008 Reporting Services *(Wiley, 2008), by my friends and co-workers Paul Turley, Thiago Silva, Bryan C. Smith, and Ken Withee.*

Service Broker

Service Broker provides the framework and services to enable the creation of asynchronous, loosely coupled applications. Service Broker implements a Service Oriented 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), without some of the limitations that MSMQ has (e.g., message size). Service Broker allows developers 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 19.

Data Tier Web Services

SQL Server 2008 provides support for creating and publishing data tier objects via HTTP without the use of an Internet Information Services (IIS) server. SQL Server 2008 registers itself with the HTTP.sys listener, allowing it to respond to Web Services requests. Developers can take advantage of this by creating 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 2008 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 2008 to include a new Peer-to-Peer Topology Wizard, which allows replication nodes to be managed using a topology viewer. The process of adding and removing nodes has also been made easier in this version of SQL Server. More detail about replication can be found in Chapter 13.

Multiple Instances

As with previous versions, SQL Server 2008 provides the capability of installing multiple instances of the database application on a single computer. SQL Server 2008 can also coexist with SQL Server 2000 and SQL Server 2005 instances installed on the same server. 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 e-mail 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 e-mail but were unwilling to install unnecessary and potentially risky software on production server assets. The 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. Configuring Database Mail is covered in Chapter 8.

A Note about Notification Services

In our *Beginning SQL Server 2005 Administration* (Wiley, 2006) book, Notification Services was introduced. If you are familiar with Notification Services and have used it with SQL Server 2000 or SQL Server 2005, you might be dismayed (or overjoyed, depending on your experience) that SQL Server Notification Services *is* no more. Most of the functionality of Notification Services has been absorbed into SQL Server Reporting Services, eliminating the need for Notification Services in SQL Server 2008.

SQL Server 2008 Editions

SQL Server 2008 comes in several 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, which supports absolutely everything that SQL Server 2008 has to offer. On the other end of the spectrum is the Express Edition, which offers very limited (but still exciting) features. Each edition, with the exception of the Compact Edition, has an x64 and x86 version. The Enterprise Edition (and consequently, the Developer Edition) also supports IA64 environments.

The available editions are:

- ❏ Enterprise Edition
- ❏ Standard Edition
- ❏ Workgroup Edition
- ❏ Web Edition
- ❏ Express Edition
- ❏ Express Advanced Edition
- ❏ Developer Edition
- ❏ Compact Edition

Chapter 1: Introducing SQL Server 2008

The following table contrasts the major differences between the four main editions of SQL Server 2008. The Developer Edition includes the same feature set as the Enterprise Edition but is not licensed for production use.

Feature	Enterprise Edition	Standard Edition	Workgroup Edition	Web Edition
Failover Clustering	Yes (limited by OS)	2-node	No	No
Multi-Instance Support	50	16	16	16
Database Mirroring	Yes	Limited	Witness Server Role only	Witness Server Role only
Enhanced Availability Features	Yes	No	No	No
Table and Index Physical Partitioning	Yes	No	No	No
Policy-Based Management	Yes	Yes	Yes	Yes
T-SQL and MDX IntelliSense	Yes	Yes	Yes	No
Spatial and Location Services	Yes	Yes	Yes	Yes
Service Broker	Yes	Yes	Yes	Client only
Analysis Services	Yes	Limited	No	No
Data Mining	Yes	Limited	No	No
Reporting Services	Yes	Limited	Limited	Limited
Integration Services	Yes	Limited	Very limited	Very limited
Replication Services	Yes	Limited	Limited features available to Subscriber only	Limited features available to Subscriber only

For a complete list of supported features, consult SQL Server 2008 Books Online under the topic ''Features Supported by the Editions of SQL Server 2008.''

SQL Server Compact 3.5 SP1

There have been several iterations of the SQL Server Compact Edition, beginning with SQL Server CE, first offered in SQL Server 2000. When SQL Server 2005 was released, it was rebranded as SQL Server 2005 Mobile Edition, specifically targeting Smartphones and PDAs. The new Compact Edition enables the installation of a small SQL Server database on a mobile device or Windows platform to support a

Windows Embedded CE or Windows Mobile application, as well as supporting desktop applications that require a much smaller feature set than offered in the Express Edition.

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 keeps track of the deliveries and new packages picked up at delivery locations. Once the truck comes back to the delivery center, the mobile device could be synchronized with the central database via replication or data upload.

SQL Server 2008 Express Edition

Back in the old days, when I had to manage database systems (in the snow, uphill both ways), the Microsoft Desktop Edition (MSDE) of SQL Server was the primary client-side Database Engine. It was extremely limited and included almost no management tools (except for the command-line osql utility) but had a compelling price — free. This has since been replaced with the SQL Server 2008 Express Edition. It's still not as robust as the Standard or Enterprise Editions, but for its very low price (you can't beat free), it still contains a great deal of functionality.

What makes this edition compelling is that it is perfect for many organizations that 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, many small businesses that are not in the tech industry often don't have the benefit of having tech-savvy personnel on staff. Viable Open Source solutions like MySQL running on Linux or Windows is simply 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, SQL Server Management Studio Basic. It also supports databases up to 4 GB in size and contains much of the same functionality as the other editions. There is even an ''advanced'' edition of SQL Server 2008 Express that includes full-text search and Reporting Services (still free!).

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. Even larger organizations can take advantage of the SQL Server Express Edition by using it for smaller, departmental or business unit installations.

SQL Server 2008 Web Edition

SQL Server 2008 Web Edition is the newest entry in the SQL Server product family. The Web Edition is designed around support for Web-facing environments and applications. With support for up to four processors and no limits on memory or database size, the Web Edition positions itself as a cost-efficient means for hosting services that rely on SQL Server databases. The Web Edition is targeted toward service providers, or Select Licensing customers that need to host public data.

Chapter 1: Introducing SQL Server 2008

The Web Edition has some very specific licensing guidelines and requirements. For example, it is licensed for public-facing web applications, sites, and services. It is not licensed for internal line-of-business applications. Because it is designed around public consumption, Client Access Licenses (CALs) are not applicable to the Web Edition.

The Web Edition is only available to Select Licensing and Service Provider License Agreement Customers. Contact your reseller or licensing representative to find out if the Web Edition is appropriate for your scenario.

SQL Server 2008 Workgroup Edition

The Workgroup Edition contains all the functionality of the SQL Server 2008 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 Editions.

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 2008 Standard Edition

Most of the capabilities of SQL Server 2008 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 features of the Analysis Services and Integration Services engines.

SQL Server 2008 Enterprise Edition

The Enterprise Edition is the bee's knees. 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 will allow you to dynamically add memory and CPUs, Transparent Data Encryption, and parallel index operations, this release is for you. It is also an appropriate solution if you require only advanced business analytics, and not necessarily the millions of transactions per second that this edition offers.

The Enterprise Edition is about performance and scalability. Although the feature set in the Enterprise Edition may be more than the average company needs, the differences in performance between the Standard and Enterprise editions can have a significant impact on whether their SQL Server is scalable

10

enough to accommodate quick growth within the organization. The Enterprise Edition fully optimizes Read-ahead execution and table scans, which results in a noticeable performance improvement.

The difference in cost between the Standard Edition and the Enterprise Edition can be significant; especially to smaller organizations where budget constraints can limit their purchasing power. However, be aware that some software may depend on certain features of the Enterprise Edition. A good example of this is the Microsoft Office PerformancePoint Server 2007 Planning Server, which relies heavily on proactive caching for Analysis Services cubes. This feature is only available in the Enterprise Edition of SQL Server.

SQL Server 2008 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 2008, the services SQL Server 2008 offers, and the types of databases used by SQL Server. This section also introduces at a high level how those databases are stored and accessed, but you can find a detailed description of the SQL Server 2008 storage architecture in Chapter 4.

SQL Server 2008 Communication

To adequately plan for a SQL Server database application, it is important to understand how SQL Server 2008 communicates with clients. As mentioned previously, SQL Server 2008 is more than just a relational database server. Because the SQL Server 2008 platform offers several different data services, it also must provide different ways of accessing that data.

SQL Server 2008 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 allow local connections without having to incur the overhead of a network protocol. A more complete description of the protocols that can be leveraged by SQL Server 2008 is provided in Chapter 7.

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, SQL Server uses 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 used 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 2008 supports native Data

Chapter 1: Introducing SQL Server 2008

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 2008 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 T-SQL with Microsoft SQL Server 2005 and 2008* by Paul Turley and Dan Wood (Wiley, 2008).

❑ **Extensible Markup Language (XML)** — This is fully supported in SQL Server 2008 as a data type, as well as language extensions to XML that enable the retrieval and modification of data by using XQuery syntax or native XML methods.

❑ **Multidimensional Expressions (MDX)** — This language is used to query against multidimensional objects in SQL Server 2008 Analysis Services.

❑ **Data Mining Extensions (DMX)** — This is an extension of Transact-SQL that enables the creation of queries against a data-mining model implemented in SQL Server 2008 Analysis Services.

❑ **Extensible Markup Language for Analysis (XMLA)** — This can be used to both discover metadata from an instance of SQL Server 2008 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 2008 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 2008, 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 Distributed Management Objects (DMOs)** — SQL-DMO is a legacy set of management objects that have been held over from SQL Server 2000. Although they are available in SQL Server 2005 and SQL Server 2008, they have been deprecated in favor of SQL-SMO. Applications that still use SQL-DMO should be upgraded to support SQL-SMO for SQL Server 2008.

SQL Server 2008 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. These services are identified by their executable names.

MSSQLServer (SQL Server)

The MSSQLServer service is the Database Engine. To connect and transact against a SQL Server 2008 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 or Windows Server 2008, it can also be configured to run under the Network System account.

SQLServerAgent (SQL Server Agent)

The SQLServerAgent service is responsible for the execution of scheduled jobs such as 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 when a job that executes perfectly during a manual invocation fails when run by the agent. Often, 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.

MSSQLServerADHelper100 (SQL Server Active Directory Helper)

Microsoft SQL Server 2008 has the ability to publish itself and its features in Active Directory. This can make it easier for Active Directory–aware services and applications to find the necessary SQL components that they need. Typically, the MSSQLServer service and the SQLServerAgent service are configured to run with a domain account that has local administrative rights on the server that 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 publish objects in 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 MSSQLServer-ADHelper service was created. The MSSQLServerADHelper service runs under the local system account of the domain computer that 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.

13

Chapter 1: Introducing SQL Server 2008

Regardless of the number of installed instances, there is only one MSSQLServerADHelper service per computer.

The version information ''100'' is used to denote that this service is associated with SQL Server 2008, or SQL Server 10.0.

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 2008's OLAP needs, as well as the Data Mining engine included with SQL Server 2008.

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 start-up, 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 the SQL Server 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 2008 Books Online.

MSSQLFDLauncher (SQL Full-Text Filter Daemon Launcher)

The Microsoft Full-Text Daemon Launcher for SQL Server (MSSQLFDLauncher) 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 MSSQLFDLauncher 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 MSSQLFDLauncher 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 T-SQL with Microsoft SQL Server 2005 and 2008*.

MSDTSServer100 (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.

ReportingServicesServer (SQL Server Reporting Services)

The ReportingServicesServer 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 Microsoft SQL Server 2008 Reporting Services*.

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 file system backups of volumes containing SQL Server log or data files will typically fail to properly back up those files, 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 of a snapshot copy of the files 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 uses a protocol known as *two-phased commit (2 PC)* to ensure that all transactions that span systems are committed on all participating systems.

SQL Server 2008 Database Objects

SQL Server 2008 database objects exist within a defined scope and hierarchy. This hierarchy enables more control over security permissions and organization of objects by similar function. SQL Server 2008 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 2008 is *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. And let's not even get started on how virtualization mucks up the term. 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 2008 Data Platform application on a single computer running a Windows operating system. Although 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 2008 and you read ''server,'' it is important to check the context to make sure that it means an instance of SQL Server 2008 or the physical computer that SQL Server is installed on.

When it comes to the server scope and SQL Server 2008 database objects, the term *server* actually refers to the SQL Server 2008 *instance*. The default instance is actually SERVERNAME\MSSQLService. However, since it is the *default instance*, appending *MSSQLService* to the server name is unnecessary. For example, we are using a server called *AUGHTEIGHT* that runs the Windows Server 2008 Operating System while writing this book. The default instance of SQL Server is known simply as *AUGHTEIGHT*. If you were to install a second instance, named *SECONDINSTANCE*, the SQL Server name would be *AUGHTEIGHT\SECONDINSTANCE*. From a SQL Server point of view, each instance is considered a separate ''server.''

Database

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

The ANSI synonym for *database* is *catalog*. When connecting to an instance of SQL Server 2008, it is generally desired to specify an Initial Catalog, or Initial Database. An instance of SQL Server 2008 can contain many databases. It used to be common for a typical database application to be constrained within one database that contained all the data objects required to provide the functionality for the application. However, now it is not uncommon to see more and more applications requiring multiple databases to manage different components of the application (this tends to increase the scalability of said application). An example of this is SharePoint, which creates databases for managing the SharePoint environment itself, as well as content databases for the various sites and site collections.

Schema

Each database can contain one or more schemas. A schema is a namespace for database objects. All data objects in a SQL Server 2008 database reside in a specific schema.

SQL Server 2008 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, and it wasn't uncommon to see everything in the database in the dbo schema. In SQL Server 2008, 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 itself. 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 2008 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 AUGHTEIGHT, and his database context is set to AdventureWorks2008. Because Fred was not assigned a user-defined schema, he exists in the default dbo schema. Fred wants to retrieve the contents of the Person table, so he executes the following query:

```
SELECT * FROM Person;
```

Fred's query will resolve to AUGHTEIGHT.AdventureWorks2008.dbo.Person. Unfortunately, that table does not exist. The fully qualified name for the contact table is AUGHTEIGHT.AdventureWorks2008 .Person.Person. 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, as in the following example:

```
SELECT * FROM Person.Person;
```

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 AdventureWorks2008;
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.DatabaseLog so he executes the following:

```
SELECT * FROM DatabaseLog;
```

SQL Server first resolves this query as AUGHTEIGHT.AdventureWorks2008.Person.DatabaseLog because Fred's default schema is Person and he did not explicitly tell SQL Server what schema to work with. Because the DatabaseLog table does not exist in the Person schema, the initial resolution fails, but SQL Server then falls back to the dbo schema and resolves the name as AUGHTEIGHT.AdventureWorks2008.dbo.DatabaseLog. The resolution succeeds, and Fred is able to retrieve 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 (e.g., 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 application's 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 (at least) a two-part name to avoid this confusion.

SQL Server 2008 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 (sometimes during the process of installing an application) who have the appropriate level of permissions to store application data.

System Databases

The system databases are comprised of `master`, `model`, `msdb`, and `tempdb` databases, as well as 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 2008. 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). SQL Server 2008 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, 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 make it easier to perform this task with the least amount of effort, a custom `TRIM` function can be added to 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 and can be used with the following simplified code:

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

Sure, it's only a small savings, but the open and close parenthesis characters 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 it will be available in subsequently created databases. This includes database users, roles, tables, stored procedures, functions, and assemblies.

The `msdb` Database

The `msdb` database can be considered 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 use 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.

One of the primary functions of this database is 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 stores 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. It is also used extensively to support 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 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 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 system objects.

Chapter 1: Introducing SQL Server 2008

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. Unlike previous versions, SQL Server 2008 does not ship with any sample databases. Instead, sample databases are available from Microsoft's Open Source CodePlex site (`www.codeplex.com`). There you can search for the three sample databases that are available at the time of this writing: `AdventureWorks2008`, `AdventureWorksLT2008`, and AdventureWorksDW2008.

The `AdventureWorks2008` database is an OLTP database used by the fictitious Adventure Works Cycles Company, which sells mountain bikes and mountain-biking-related merchandise.

The `AdventureWorksLT2008` database is an OLTP database that is a subset of the larger `AdventureWorks2008` database. It was scaled down to help those who are new to relational databases.

The `AdventureWorksDW2008` database is an OLAP database used for data analysis of historical Adventure Works Cycles data.

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.

A 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 2008 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>\MSSQL10.MSSQLSERVER\MSSQL\Data\master.mdf <install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\mastlog.ldf
model	<install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\model.mdf <install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\modellog.ldf
msdb	<install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\msdbdata.mdf <install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\msdblog.ldf
tempdb	<install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\tempdb.mdf <install path>\MSSQL10.MSSQLSERVER\MSSQL\Data\templog.ldf
resource	<install path>\MSSQL10.MSSQLSERVER\MSSQL\Binn\Mssqlsystemresource.mdf <install path>\MSSQL10.MSSQLSERVER\MSSQL\Binn\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 2008 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 has been almost completely removed from SQL Server 2008. 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, which consists 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).

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 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.
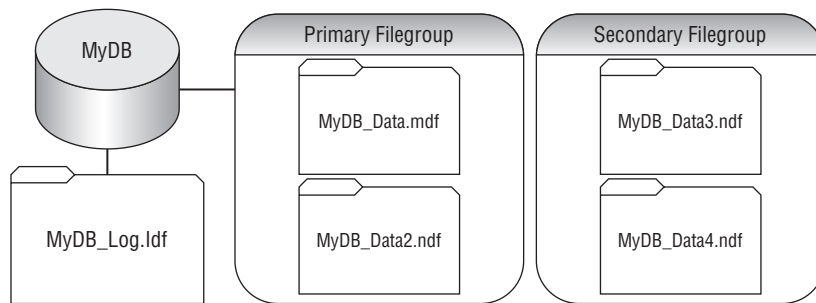

Figure 1-1: Data files and filegroups.

SQL Server Security

Chapter 6 provides a thorough discussion of SQL Server 2008 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 2008 can be configured to work in either the Windows Authentication Mode or the SQL Server and Windows Authentication Mode, which is 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 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. This is the default behavior of a fresh installation of SQL Server 2008.

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.

Summary

This chapter introduced the basic structure and purpose of SQL Server 2008, 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 will learn how to plan and perform a SQL Server 2008 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.