Chapter 1

The World of SQL Server

COPYRIGHTED MATERIAL

Welcome to SQL Server 2008.

At the Rocky Mountain Tech Tri-Fecta 2009 SQL keynote, I walked through the major SQL Server 2008 new features and asked the question, "Cool or Kool-Aid?"

I've worked with SQL Server since version 6.5 and I'm excited about this newest iteration because it reflects a natural evolution and maturing of the product. I believe it's the best release of SQL Server so far. There's no Kool-Aid here — it's all way cool.

SQL Server is a vast product and I don't know any sane person who claims to know all of it in depth. In fact, SQL Server is used by so many different types of professions to accomplish so many different types of tasks, it can be difficult to concisely define it, but here goes:

SQL Server 2008: Microsoft's enterprise client-server relational database product, with T-SQL as its primary programming language.

However, SQL Server is more than just a relational database engine:

- Connecting to SQL Server is made easy with a host of data connectivity options and a variety of technologies to import and export data, such as Tabular Data Stream (TDS), XML, Integration Services, bulk copy, SQL Native Connectivity, OLE DB, ODBC, and distributed query with Distributed Transaction Coordinator, to name a few.
- The engine works well with data (XML, spatial, words within text, and blob data).
- SQL Server has a full suite of OLAP/BI components and tools to work with multidimensional data, analyze data, build cubes, and mine data.

IN THIS CHAPTER

- SQL Server includes a complete reporting solution that serves up great-looking reports, enables users to create reports, and tracks who saw what when.
- SQL Server exposes an impressive level of diagnostic detail with Performance Studio, SQL Trace/Profiler, and Database Management Views and Functions.
- SQL Server includes several options for high availability with varying degrees of latency, performance, number of nines, physical distance, and synchronization.
- SQL Server can be managed declaratively using Policy-Based Management.
- SQL Server's Management Studio is a mature UI for both the database developer and the DBA.
- SQL Server is available in several different scalable editions in 32-bit and 64-bit for scaling up and out.

All of these components are included with SQL Server (at no additional cost or per-component cost), and together in concert, you can use them to build a data solution within a data architecture environment that was difficult or impossible a few years ago. SQL Server 2008 truly is an enterprise database for today.

A Great Choice

There are other good database engines, but SQL Server is a great choice for several reasons. I'll leave the marketing hype for Microsoft; here are my personal ten reasons for choosing SQL Server for my career:

- **Set-based SQL purity:** As a set-based data architect type of guy, I find SQL Server fun to work with. It's designed to function in a set-based manner. Great SQL Server code has little reason to include iterative cursors. SQL Server is pure set-based SQL.
- **Scalable performance:** SQL Server performance scales well — I've developed code on my notebook that runs great on a server with 32- and 64-bit dual-core CPUs and 48GB of RAM. I've yet to find a database application that doesn't run well with SQL Server given a good design and the right hardware.

  People sometimes write to the newsgroups that their database is huge — *"over a gig!"* — but SQL Server regularly runs databases in the terabyte size. I'd say that over a petabyte is huge, over a terabyte is large, 100 GB is normal, under 10 GB is small, and under 1 GB is tiny.
- **Scalable experience:** The SQL Server experience scales from nearly automated self-managed databases administered by the accidental DBA to finite control that enables expert DBAs to tune to their heart's content.
- **Industry acceptance:** SQL Server is a standard. I can find consulting work from small shops to the largest enterprises running SQL Server.
- **Diverse technologies:** SQL Server is broad enough to handle many types of problems and applications. From BI to spatial, to heavy transactional OLTP, to XML, SQL Server has a technology to address the problem.
- **SQL in the Cloud:** There are a number of options to host a SQL database in the cloud with great stability, availability, and performance.
- **Financial stability:** It's going to be here for a nice long time. When you choose SQL Server, you're not risking that your database vendor will be gone next year.

- **Ongoing development:** I know that Microsoft is investing heavily in the future of SQL Server, and new versions will keep up the pace of new cool features. I can promise you that SQL 11 will rock!
- **Fun community:** There's an active culture around SQL Server, including a lot of user groups, books, blogs, websites, code camps, conferences, and so on. Last year I presented 22 sessions at nine conferences, so it's easy to find answers and get plugged in. In fact, I recently read a blog comparing SQL Server and Oracle and the key differentiator is enthusiasm of the community and the copious amount of information it publishes. It's true: the SQL community is a fun place to be.
- **Affordable:** SQL Server is more affordable than the other enterprise database options, and the Developer Edition costs less than $50 on Amazon.

The Client/Server Database Model

Technically, the term *client/server* refers to any two cooperating processes. The client process requests a service from the server process, which in turn handles the request for the client. The client process and the server process may be on different computers or on the same computer: It's the cooperation between the processes that is significant, not the physical location.

For a client/server database, the client application (be it a front end, an ETL process, a middle tier, or a report) prepares a SQL request — just a small text message or remote procedure call (RPC) — and sends it to the database server, which in turn reads and processes the request. Inside the server, the security is checked, the indexes are searched, the data is retrieved or manipulated, any server-side code is executed, and the final results are sent back to the client. All the database work is performed within the database server. The actual data and indexes never leave the server.

In contrast, desktop file-based databases (such as Microsoft Access), may share a common file, but the desktop application does all the work as the data file is shared across the network.

The client/server–database model offers several benefits over the desktop database model:

- Reliability is improved because the data is not spread across the network and several applications. Only one process handles the data.
- Data integrity constraints and business rules can be enforced at the server level, resulting in a more thorough implementation of the rules.
- Security is improved because the database keeps the data within a single server. Hacking into a data file that's protected within the database server is much more difficult than hacking into a data file on a workstation. It's also harder to steal a physical storage device connected to a server, as most server rooms are adequately protected against intruders.
- Performance is improved and better balanced among workstations because the majority of the workload, the database processing, is being handled by the server; the workstations handle only the user-interface portion.

continued

continued

- Because the database server process has direct access to the data files, and much of the data is already cached in memory, database operations are much faster at the server than in a multi-user desktop-database environment. A database server is serving every user operating a database application; therefore, it's easier to justify the cost of a beefier server. For applications that require database access and heavy computational work, the computational work can be handled by the application, further balancing the load.
- Network traffic is greatly reduced. Compared to a desktop database's rush-hour traffic, client/server traffic is like a single motorcyclist carrying a slip of paper with all 10 lanes to himself. This is no exaggeration! Upgrading a heavily used desktop database to a well-designed client/server database will reduce database-related network traffic by more than 95 percent.
- A by-product of reducing network traffic is that well-designed client/server applications perform well in a distributed environment — even when using slower communications. So little traffic is required that even a 56KB dial-up line should be indistinguishable from a 100baseT Ethernet connection for a .NET-rich client application connected to a SQL Server database.

  Client/server SQL Server: a Boeing 777. Desktop databases: a toy red wagon.

SQL Server Database Engine

SQL Server components can be divided into two broad categories: those within the engine, and external tools (e.g., user interfaces and components), as illustrated in Figure 1-1. Because the relational Database Engine is the core of SQL Server, I'll start there.

Database Engine

The SQL Server *Database Engine*, sometimes called the *Relational Engine*, is the core of SQL Server. It is the component that handles all the relational database work. SQL is a descriptive language, meaning it describes only the question to the engine; the engine takes over from there.

Within the Relational Engine are several key processes and components, including the following:

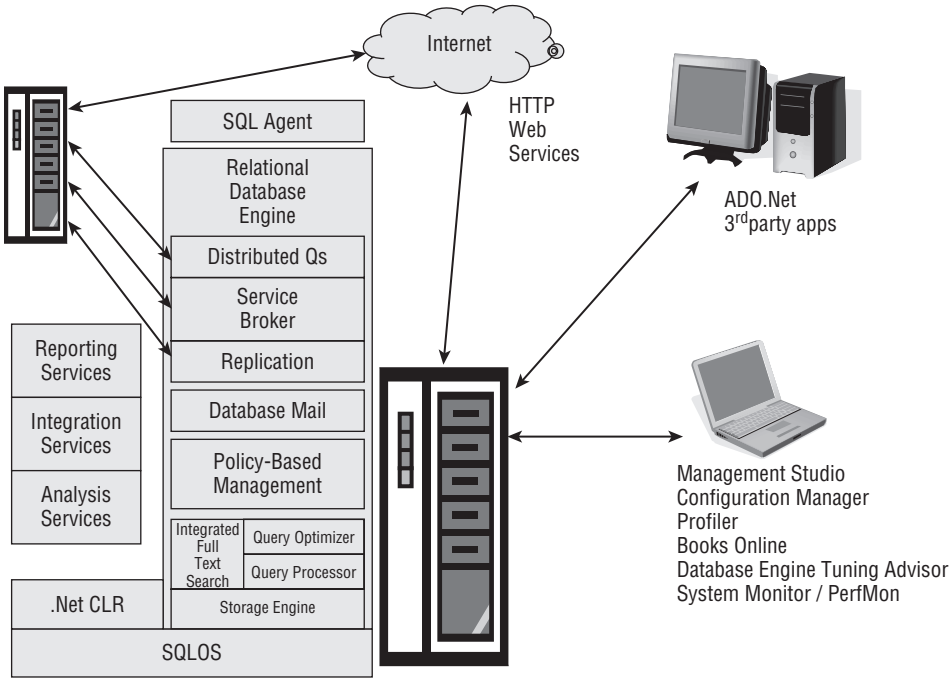
- **Algebrizer:** Checks the syntax and transforms a query to an internal representation that is used by the following components.
- **Query Optimizer:** SQL Server's Query Optimizer determines how to best process the query based on the costs of different types of query-execution operations. The estimated and actual query-execution plans may be viewed graphically, or in XML, using Management Studio or SQL Profiler.
- **Query Engine, or Query Processor:** Executes the queries according to the plan generated by the Query Optimizer.

- **Storage Engine:** Works for the Query Engine and handles the actual reading from and writing to the disk.
- **The Buffer Manager:** Analyzes the data pages being used and pre-fetches data from the data file(s) into memory, thus reducing the dependency on disk I/O performance.
- **Checkpoint:** Process that writes dirty data pages (modified pages) from memory to the data file.
- **Resource Monitor:** Optimizes the query plan cache by responding to memory pressure and intelligently removing older query plans from the cache.
- **Lock Manager:** Dynamically manages the scope of locks to balance the number of required locks with the size of the lock.
- **SQLOS:** SQL Server eats resources for lunch, and for this reason it needs direct control of the available resources (memory, threads, I/O request, etc.). Simply leaving the resource management to Windows isn't sophisticated enough for SQL Server. SQL Server includes its own OS layer, *SQLOS*, which manages all of its internal resources.

SQL Server 2008 supports installation of up to 16 (Workgroup Edition) or 50 (Standard or Enterprise Edition) instances of the Relational Engine on a physical server. Although they share some components, each instance functions as a complete separate installation of SQL Server.

FIGURE 1-1

SQL Server is a collection of components within the relational Database Engine and client components.



ACID and SQL Server's Transaction Log

SQL Server's Transaction Log is more than an optional appendix to the engine. It's integral to SQL Server's reputation for data integrity and robustness. Here's why:

Data integrity is defined by the acronym ACID, meaning transactions must be Atomic (one action — all or nothing), Consistent (the database must begin and end the transaction in a consistent state), Isolated (no transaction should affect another transaction), and Durable (once committed, always committed).

The transaction log is vital to the ACID capabilities of SQL Server. SQL Server writes to the transaction log as the first step of writing any change to the data pages (in memory), which is why it is sometimes called the *write-ahead transaction log*.

Every DML statement (Select, Insert, Update, Delete) is a complete transaction, and the transaction log ensures that the entire set-based operation takes place, thereby ensuring the atomicity of the transaction.

SQL Server can use the transaction log to roll back, or complete a transaction regardless of hardware failure, which is key to both the consistency and durability of the transaction.

CROSS-REF Chapter 40, "Policy-Based Management," goes into more detail about transactions.

Transact-SQL

SQL Server is based on the SQL standard, with some Microsoft-specific extensions. SQL was invented by E. F. Codd while he was working at the IBM research labs in San Jose in 1971. SQL Server is entry-level (Level 1) compliant with the ANSI SQL 92 standard. (The complete specifications for the ANSI SQL standard are found in five documents that can be purchased from `www.techstreet.com/ncits.html`. I doubt if anyone who doesn't know exactly what to look for will find these documents.) But it also includes many features defined in later versions of the standard (SQL-1999, SQL-2003).

While the ANSI SQL definition is excellent for the common data-selection and data-definition commands, it does not include commands for controlling SQL Server properties, or provide the level of logical control within batches required to develop a SQL Server–specific application. Therefore, the Microsoft SQL Server team has extended the ANSI definition with several enhancements and new commands, and has left out a few commands because SQL Server implemented them differently. The result is Transact-SQL, or T-SQL — the dialect of SQL understood by SQL Server.

Missing from T-SQL are very few ANSI SQL commands, primarily because Microsoft implemented the functionality in other ways. T-SQL, by default, also handles nulls, quotes, and padding differently than the ANSI standard, although that behavior can be modified. Based on my own development experience, I can say that none of these differences affect the process of developing a database application using SQL Server. T-SQL adds significantly more to ANSI SQL than it lacks.

Understanding SQL Server requires understanding T-SQL. The native language of the SQL Server engine is Transact-SQL. Every command sent to SQL Server must be a valid T-SQL command. Batches of stored T-SQL commands may be executed within the server as stored procedures. Other tools, such as Management Studio, which provide graphical user interfaces with which to control SQL Server, are at some level converting most of those mouse clicks to T-SQL for processing by the engine.

SQL and T-SQL commands are divided into the following three categories:

- **Data Manipulation Language (DML):** Includes the common SQL `SELECT`, `INSERT`, `UPDATE`, and `DELETE` commands. DML is sometimes mistakenly referred to as *Data Modification Language*; this is misleading, because the `SELECT` statement does not modify data. It does, however, manipulate the data returned.
- **Data Definition Language (DDL):** Commands that `CREATE`, `ALTER`, or `DROP` data tables, constraints, indexes, and other database objects.
- **Data Control Language (DCL):** Security commands such as `GRANT`, `REVOKE`, and `DENY` that control how a principal (user or role) can access a securable (object or data.)

In Honor of Dr. Jim Gray

Jim Gray, a Technical Fellow at Microsoft Research (MSR) in San Francisco, earned the ACM Turing Award in 1998 ''for seminal contributions to database and transaction processing research and technical leadership in system implementation.''

A friend of the SQL Server community, he often spoke at PASS Summits. His keynote address on the future of databases at the PASS 2005 Community Summit in Grapevine, Texas, was one of the most thrilling database presentations I've ever seen. He predicted that the exponential growth of cheap storage space will create a crisis for the public as they attempt to organize several terabytes of data in drives that will fit in their pockets. For the database community, Dr. Gray believed that the growth of storage space would eliminate the need for updating or deleting data; future databases will only have insert and select commands.

The following image of Microsoft's TerraServer appeared in the SQL Server 2000 Bible. TerraServer was Microsoft's SQL Server 2000 scalability project, designed by Jim Gray. His research in spatial data is behind the new spatial data types in SQL Server 2008.

Dr. Gray's research led to a project that rocked the SQL world: 48 SATA drives were configured to build a 24-TB data warehouse, achieving throughputs equal to a SAN at one-fortieth the cost.

On January 28, 2007, Jim Gray disappeared while sailing alone near the Farallon Islands just outside San Francisco Bay. An extensive sea search by the U.S. Coast Guard, a private search, and an Internet satellite image search all failed to reveal any clues.

continued

continued

Policy-Based Management

Policy-based management, affectionately known as PBM, and new for SQL Server 2008, is the system within the engine that handles declarative management of the server, database, and any database object.

As declarative management, PBM replaces the chaos of scripts, manual operations, three-ring binders with daily run sheets and policies, and Post-it notes in the DBA's cubicle.

CROSS-REF **Chapter 40, "Policy-Based Management," discusses managing your server declaratively.**

.NET Common Language Runtime

Since SQL Server 2005, SQL Server has hosted an internal .Net Common Language Runtime, or CLR. Assemblies developed in Visual Studio can be deployed and executed inside SQL Server as stored procedures, triggers, user-defined functions, or user-defined aggregate functions. In addition, data types developed with Visual Studio can be used to define tables and store custom data.

SQL Server's internal operating system, SQLOS, actually hosts the .NET CLR inside SQL Server. There's value in SQLOS hosting the CLR, as it means that SQL Server is in control of the CLR resources. It can prevent a CLR problem, shut down and restart a CLR routine that's causing trouble, and ensure that the battle for memory is won by the right player.

While the CLR may sound appealing, Transact-SQL is the native language of SQL Server and it performs better and scales better than the CLR for nearly every task. The CLR is useful for coding tasks that

require resources external to the database that cannot be completed using T-SQL. In this sense, the CLR is the replacement for the older extended stored procedures. In my opinion, the primary benefit of the CLR is that Microsoft can use it to extend and develop SQL Server.

By default, the common language runtime is disabled in SQL Server and must be specifically enabled using a T-SQL SET command. When enabled, each assembly's scope, or ability to access code outside SQL Server, can be carefully controlled.

CROSS-REF .NET integration is discussed in Chapter 32, "Programming .NET CLR within SQL Server."

Service Broker

Introduced in SQL Server 2005, Service Broker is a managed data queue, providing a key performance and scalability feature by leveling the load over time:

- Service Broker can buffer high volumes of calls to an HTTP Web Service or a stored procedure. Rather than a thousand Web Service calls launching a thousand stored procedure threads, the calls can be placed on a queue and the stored procedures can be executed by a few instances to handle the load more efficiently.
- Server-side processes that include significant logic or periods of heavy traffic can place the required data in the queue and return to the calling process without completing the logic. Service Broker will move through the queue calling another stored procedure to do the heavy lifting.

While it's possible to design your own queue within SQL Server, there are benefits to using Microsoft's work queue. SQL Server includes DDL commands to manage Service Broker, and there are T-SQL commands to place data on the queue or fetch data from the queue. Information about Service Broker queues are exposed in metadata views, Management Studio, and System Monitor. Most important, Service Broker is well tested and designed for heavy payloads under stress.

CROSS-REF Service Broker is a key service in building a service-oriented architecture data store. For more information, see Chapter 35, "Building Asynchronous Applications with Service Broker."

Replication services

SQL Server data is often required throughout national or global organizations, and SQL Server replication is often employed to move that data. Replication Services can move transactions one-way or merge updates from multiple locations using a publisher-distributor-subscriber topology.

CROSS-REF Chapter 36, "Replicating Data," explains the various replication models and how to set up replication.

Integrated Full-Text Search

Full-Text Search has been in SQL Server since version 7, but with each version this excellent service has been enhanced, and the name has evolved to Integrated Full-Text Search, or iFTS.

SQL queries use indexes to locate rows quickly. SQL Server b-tree indexes index the entire column. Searching for words within the column requires a scan and is a very slow process. Full-Text Search solves this problem by indexing every word within a column.

Once the full-text search has been created for the column, SQL Server queries can search the Full-Text Search indexes and return high-performance in-string word searches.

CROSS-REF Chapter 19, "Using Integrated Full-Text Search," explains how to set up and use full-text searches within SQL queries.

Server management objects

Server Management Objects (SMO) is the set of objects that exposes SQL Server's configuration and management features for two primary purposes: scripting and programming.

For administration scripting, PowerShell cmdlets use SMO to access SQL Server objects.

For programming, SMO isn't intended for development of database applications; rather, it's used by vendors when developing SQL Server tools such as Management Studio or a third-party management GUI or backup utility. SMO uses the namespace `Microsoft.SQLServer.SMO`.

Filestream

New for SQL Server 2008, Filestream technology adds the ability to write or read large BLOB data to Windows files through the database, complete with transactional control.

CROSS-REF Chapter 15, "Modifying Data," covers how to use FileStream with `INSERT`, `UPDATE`, and `DELETE` commands.

SQL Server Services

The following components are client processes for SQL Server used to control, or communicate with, SQL Server.

SQL Server Agent

The Server Agent is an optional process that, when running, executes SQL jobs and handles other automated tasks. It can be configured to automatically run when the system boots, or it can be started from SQL Server Configuration Manager or Management Studio's Object Explorer.

Database Mail

The Database Mail component enables SQL Server to send mail to an external mailbox through SMTP. Mail may be generated from multiple sources within SQL Server, including T-SQL code, jobs, alerts, Integration Services, and maintenance plans.

CROSS-REF Chapter 43, "Automating Database Maintenance with SQL Server Agent," details SQL agents and jobs, as well as the SQL Server Agent. It also explains how to set up a mail profile for SQL Server and how to send mail.

Distributed Transaction Coordinator (DTC)

The Distributed Transaction Coordinator is a process that handles dual-phase commits for transactions that span multiple SQL Servers. DTC can be started from within Windows' Computer Administration/Services. If the application regularly uses distributed transactions, you should start DTC when the operating system starts.

CROSS-REF Chapter 31, "Executing Distributed Queries," explains dual-phase commitments and distributed transactions.

A Brief History of SQL Server

SQL Server has grown considerably over the past two decades from its early roots with Sybase:

SQL Server 1.0 was jointly released in 1989 by Microsoft, Sybase, and Ashton-Tate. The product was based on Sybase SQL Server 3.0 for Unix and VMS.

SQL Server 4.2.1 for Windows NT was released in 1993. Microsoft began making changes to the code.

SQL Server 6.0 (code-named SQL 95) was released in 1995. In 1996, the 6.5 upgrade (Hydra) was released. It included the first version of Enterprise Manager (StarFighter I) and SQL Server Agent (StarFighter II).

SQL Server 7.0 (Sphinx) was released in 1999, and was a full rewrite of the Database Engine by Microsoft. From a code perspective, this was the first Microsoft SQL Server. SQL Server 7 also included English Query (Argo), OLAP Services (Plato), replication, Database Design and Query tools (DaVinci) and Full-Text Search (aptly code-named Babylon.) Data Transformation Services (DTS) is also introduced.

My favorite new feature? DTS.

SQL Server 2000 (Shiloh) 32-bit, version 8, introduced SQL Server to the enterprise with clustering, much better performance, and real OLAP. It supported XML though three different XML add-on packs. It added user-defined functions, indexed views, clustering support, Distributed Partition Views, and improved replication. SQL Server 2000 64-bit version for Intel Itanium (Liberty) was released in 2003, along with the first version of Reporting Services (Rosetta) and Data Mining tools (Aurum). DTS became more powerful and gained in popularity. Northwind joined Pubs as the sample database.

My favorite new feature? User-defined functions.

SQL Server 2005 (Yukon), version 9, was another rewrite of the Database Engine and pushed SQL Server further into the enterprise space. 2005 added a ton of new features and technologies, including Service Broker, Notification Services, CLR, XQuery and XML data types, and SQLOS. T-SQL gained try-catch and the system tables were replaced with Dynamic Management Views (DMVs). Management Studio replaced Enterprise Manager and Query Analyzer. DTS is replaced by Integration Services. English Query was removed, and stored procedure debugging was moved from the DBA interface to Visual Studio. AdventureWorks and AdventureWorksDW replaced Northwind and Pubs as the sample databases. SQL Server 2005 supported 32-bit, 64x, and Itanium CPUs. Steve Ballmer publically vowed to never again make customers wait five years between releases, and to return to a 2–3 year release cycle.

My favorite new features? T-SQL Try-Catch, Index Include columns, VarChar(max), windowing/ranking functions, and DMVs.

SQL Server 2008 (Katmai), version 10, is a natural evolution of SQL Server, adding Policy-Based Management, data compression, Resource Governor, and new beyond relational data types. Notification Services go the way of English Query. T-SQL finally gets date and time data types and table-valued parameters, the debugger returns, and Management Studio gets IntelliSense.

My favorite new features? Table-valued parameters and policy-based management.

continued

continued
What's next?

SQL Data Services is Microsoft's database in the cloud.

Kilimanjaro, estimated availability in mid-2010 extends SQL Server's BI suite with tighter integration with Office 14.

SQL11 continues the strategic direction of SQL Server 2008.

Business Intelligence

Business intelligence (BI) is the name given to the discipline and tools that enables the management of data for the purpose of analysis, exploration, reporting, mining, and visualization. While aspects of BI appear in many applications, the BI approach and toolset provides a rich and robust environment to understand data and trends.

SQL Server provides a great toolset to build BI applications, which explains Microsoft's continued gains in the growing BI market. SQL Server includes three services designed for business intelligence: Integration Services (IS, or sometimes called SSIS for SQL Server Integration Services), Reporting Services (RS), and Analysis Services (AS). Development for all three services can be done using the BI Development Studio.

Business Intelligence Development Studio

The BI Development Studio (BIDS) is a version of Visual Studio that hosts development of Analysis Services databases and mining models, Integration Services packages, and Reporting Services reports. When installed on a system that already has Visual Studio installed, these additional project types are added to the existing Visual Studio environment.

Integration Services

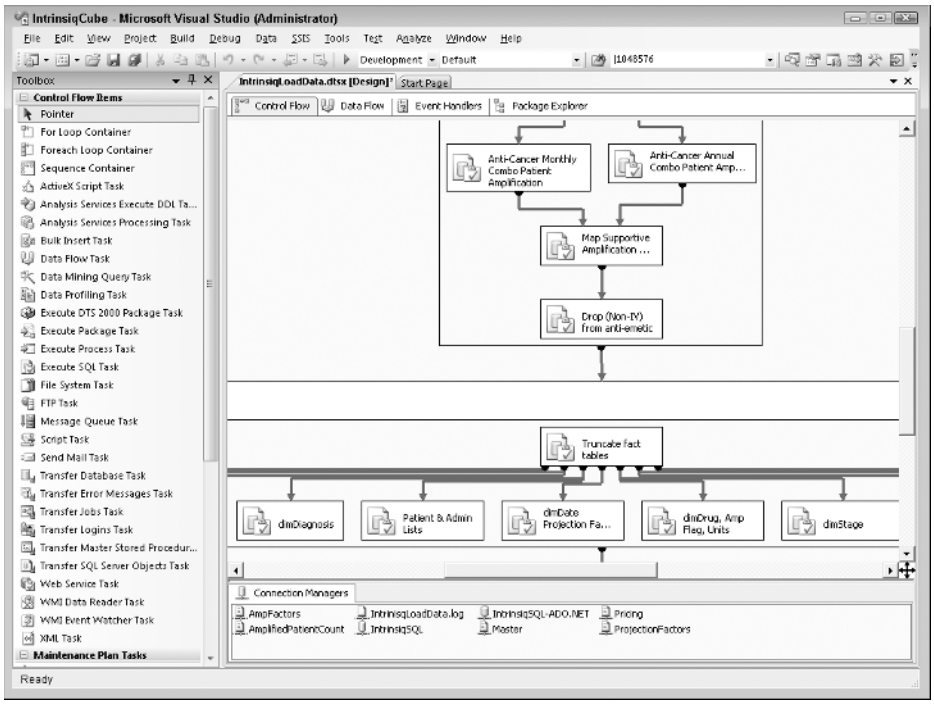
Integration Services (IS) moves data among nearly any type of data source and is SQL Server's extract-transform-load (ETL) tool. As shown in Figure 1-2, IS uses a graphical tool to define how data can be moved from one connection to another connection. Integration Services packages have the flexibility to either copy data column for column or perform complex transformations, lookups, and exception handling during the data move. Integration Services is extremely useful during data conversions, collecting data from many dissimilar data sources, or gathering for data warehousing data that can be analyzed using Analysis Services.

Integration Services has many advantages over using custom programming or T-SQL to move and transform data; chief among these are speed and traceability. If you have experience with other databases but are new to SQL Server, this is one of the tools that will most impress you. If any other company were marketing SSIS it would be their flagship product, but instead it's bundled inside SQL Server without much fanfare and at no extra charge. Be sure to find the time to explore Integration Services.

CROSS-REF **Chapter 37, "Performing ETL with Integration Services," describes how to create and execute an SSIS package.**

FIGURE 1-2

Integration Services graphically illustrates the data transformations within a planned data migration or conversion.



Analysis Services

The Analysis Services service hosts two key components of the BI toolset: *Online Analytical Processing (OLAP)* hosts multidimensional databases, whereby data is stored in cubes, while *Data Mining* provides methods to analyze data sets for non-obvious patterns in the data.

OLAP

Building cubes in a multidimensional database provides a fast, pre-interpreted, flexible analysis environment. Robust calculations can be included in a cube for later querying and reporting, going a long way toward the "one version of the truth" that is so elusive in many organizations. Results can be used as the basis for reports, but the most powerful uses involve interactive data exploration using tools such as Excel pivot tables or similar query and analysis applications. Tables and charts that summarize billions of rows can be generated in seconds, enabling users to understand the data in ways they never thought possible.

Whereas relational databases in SQL Server are queried using T-SQL, cubes are queried using Multidimensional Expressions (MDX), a set-based query language tailored to retrieving multidimensional data (see Figure 1-3). This enables relatively easy custom application development in addition to standard analysis and reporting tools.

FIGURE 1-3

Browsing a multidimensional cube within Analysis Services is a fluid way to compare various aspects of the data.

Data Mining

Viewing data from cubes or even relational queries can reveal the obvious trends and correlations in a data set, but data mining can expose the non-obvious ones. The robust set of mining algorithms enables tasks like finding associations, forecasting, and classifying cases into groups. Once a model is trained on an existing set of data, it can predict new cases that are likely to occur — for example, predicting the most profitable customers to spend scarce advertising dollars on or estimating expected component failure rates based on its characteristics.

CROSS-REF **Chapters 71, 72, and 76 cover designing cubes, data mining, and programming MDX queries with Analysis Services. Chapter 75 describes data analysis in Excel, including accessing cube data and using Excel's related data-mining functions.**

The BI Stack

The buzz today is around the BI stack; in other words, the suite of products that support an organization's BI efforts both inside and outside of SQL Server.

continued

continued

The SQL Server products described here are the BI platform of the stack, providing the data acquisition, storage, summarization, and reporting — the basis for analysis. Excel is a key analysis tool in the stack, exposing query results, pivot tables, basic data-mining functions, and a host of charting and formatting features. The 2007 version of Excel puts it on a par with many of the third-party data analysis tools available.

The PerformancePoint product enables the construction of dashboards and scorecards, exposing company performance metrics throughout the organization. Combined with a variety of reporting and charting options, key performance indicators, and data-mining trends, this product has the potential to change the way companies work day to day.

SharePoint has long been an excellent tool for collaboration, organizing and tracking documents, and searching all the associated content. Microsoft Office SharePoint Server 2007 (MOSS) adds several BI features to the SharePoint product, including Excel Services, which enables the sharing of Excel workbooks, tables, and charts via a web interface, rather than a download. MOSS also provides a great way to organize and share Reporting Services reports with related content (e.g., department reports from a department's website). In addition, BI Web Parts, SharePoint-hosted key performance indicators and Dashboards, and data connection libraries are all available within MOSS.

These additions to the BI platform provide important pieces to deliver BI solutions, presenting some very persuasive enhancements to how organizations work.

Reporting Services

Reporting Services (RS) for SQL Server 2005 is a full-featured, web-based, managed reporting solution. RS reports can be exported to PDF, Excel, or other formats with a single click, and are easy to build and customize.

Reports are defined graphically or programmatically and stored as `.rdl` files in the Reporting Services databases in SQL Server. They can be scheduled to be pre-created and cached for users, e-mailed to users, or generated by users on-the-fly with parameters. Reporting Services is bundled with SQL Server so there are no end-user licensing issues. It's essentially free, although most DBAs place it on its own dedicated server for better performance.

With SQL Server 2008, Reporting Services gets a facelift: slick new Dundas controls, a new Tablix control, a re-written memory management system, and a direct HTTP.sys access, so IIS is no longer needed. If you're still using Crystal Reports, why?

CROSS-REF **Chapters 73 and 74 deal with authoring and deploying reports using Reporting Services.**

UI and Tools

SQL Server 2008 retains most of the UI feel of SQL Server 2005, with a few significant enhancements.

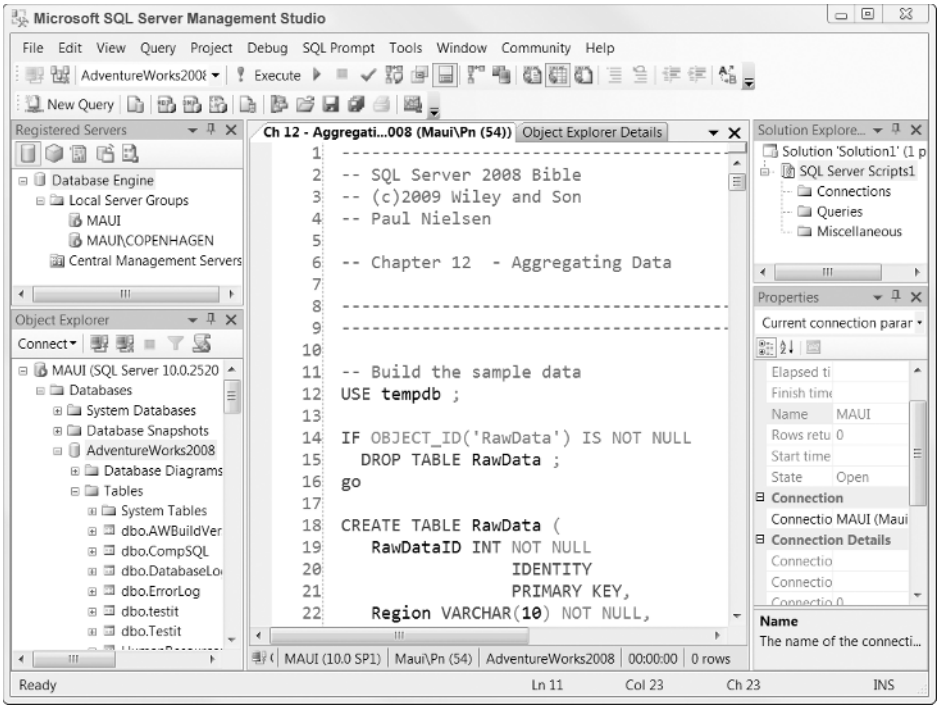
SQL Server Management Studio

Management Studio is a Visual Studio-esque integrated environment that's used by database administrators and database developers. At its core is the visual Object Explorer, complete with filters and the capability to browse all the SQL Server servers (Database Engine, Analysis Services, Reporting Services, etc.). Management Studio's Query Editor is an excellent way to work with raw T-SQL code and it's integrated with the Solution Explorer to manage projects. Although the interface, shown in Figure 1-4, can look crowded, the windows are easily configurable and can auto-hide.

CAUTION **Chapter 6, "Using Management Studio," discusses the many tools within Management Studio and how to use this flexible development and management interface.**

FIGURE 1-4

Management Studio's full array of windows and tools can seem overwhelming but it's flexible enough for you to configure it for your own purposes.

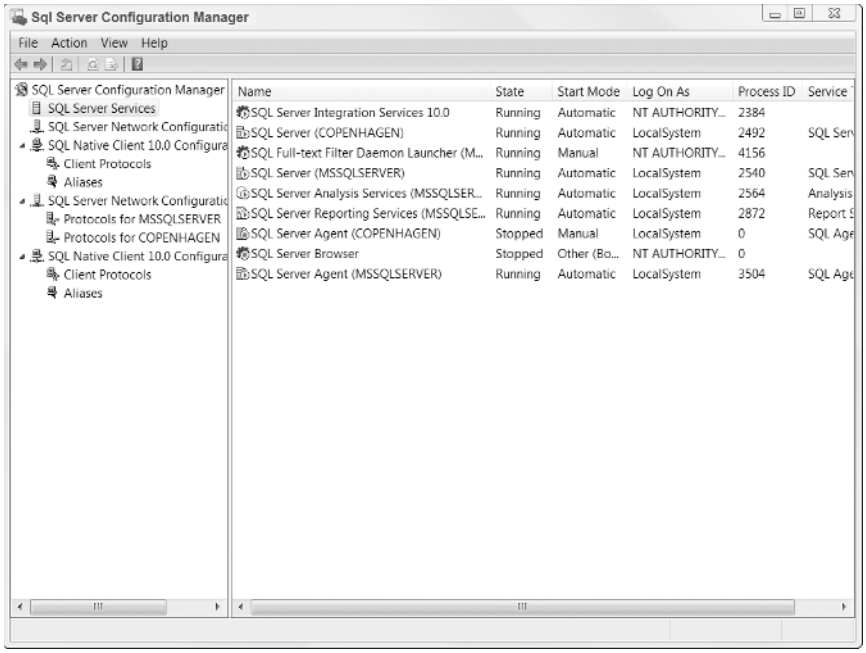


SQL Server Configuration Manager

This tool, shown in Figure 1-5, is used to start and stop any server, set the start-up options, and configure connectivity. It may be launched from the Start menu or from Management Studio.

FIGURE 1-5

Use the Configuration Manager to launch and control SQL Server's many servers.



SQL Profiler/Trace

SQL Server is capable of exposing a trace of selected events and data points. The server-side trace has nearly no load on the server. SQL Profiler is the UI for viewing traces in real time (with some performance cost) or viewing saved trace files. Profiler is great for debugging an application or tuning the database. Using a two-monitor system, I often run Profiler on one monitor while I'm tuning and developing on the other monitor.

CROSS-REF **Chapter 56, "Tracing and Profiling," covers Performance Monitor.**

Performance Monitor

While Profiler records large sets of details concerning SQL traffic and SQL Server events, Performance Monitor is a visual window into the current status of the selected performance counters. Performance Monitor is found within Windows' administrative tools. When SQL Server is installed, it adds a ton of useful performance counters to Performance Monitor. It's enough to make a network administrator jealous.

Command-line utilities

Various command-line utilities may be used to execute SQL code (sqlcmd) or perform bulk-copy operations (bcp) from the DOS prompt or a command-line scheduler. Integration Services and SQL Server

Agent have rendered these tools somewhat obsolete, but in the spirit of extreme flexibility, Microsoft still includes them.

Management Studio has a mode that enables you to use the Query Editor as if it were the command-line utility sqlcmd.

CROSS-REF **Chapter 6, "Using Management Studio," has a sidebar discussing SQLCMD and Query Editor's SQLCMD mode.**

Books Online

The SQL Server documentation team did an excellent job with Books Online (BOL) — SQL Server's mega help on steroids. The articles tend to be complete and include several examples. The indexing method provides a short list of applicable articles. BOL may be opened from Management Studio or directly from the Start menu.

BOL is well integrated with the primary interfaces. Selecting a keyword within Management Studio's Query Editor and pressing F1 will launch BOL to the selected keyword. The Enterprise Manager help buttons also launch the correct BOL topic.

Management Studio also includes a dynamic help window that automatically tracks the cursor and presents help for the current keyword.

Searching returns both online and local MSDN articles. In addition, BOL searches the Codezone community for relevant articles. Both the Community Menu and Developer Center launch web pages that enable users to ask a question or learn more about SQL Server.

CROSS-REF **Microsoft regularly updates BOL. The new versions can be downloaded from `www.Microsoft.com/sql`, and I post a link to it on `www.SQLServerBible.com`.**

NOTE **If you haven't discovered CodePlex.com, allow me to introduce it to you. CodePlex.com is Microsoft's site for open-source code. That's where you'll find AdventureWorks2008, the official sample database for SQL Server 2008, along with AdventureWorksLT (a smaller version of AdventureWorks) and AdventureWorksDW (the BI companion to AdventureWorks). You'll also find copies of the older sample databases, Pubs and Northwind. I also have a few projects on CodePlex.com myself.**

SQL Server Editions

SQL Server is available in several editions (not to be confused with versions), which differ in terms of features, hardware, and cost. This section details the various editions and their respective features. Because Microsoft licensing and costs change, check with `www.microsoft.com/sql` or your Microsoft representative for license and cost specifics.

- **Enterprise (Developer) Edition:** This is the high-end edition, with the advanced performance and availability features (e.g., table partitioning, data compression) required to support thousands of connections and databases measured by terabytes.

  The Developer Edition is the same as the Enterprise Edition, but it's licensed only for development and testing and it can run on workstation versions of Windows.

- **Standard Edition:** The majority of medium to large production database needs will be well served by the SQL Server 2008 Standard Edition, the workhorse edition of SQL Server. This edition includes all the right features, including Integration Services, Analysis Services, Web

Services, database mirroring, and failover clustering. Don't blow your budget on Enterprise Edition unless you've proven that Standard Edition won't do the job.

NOTE **With multi-core CPUs becoming commonplace in servers, the question is, how does this affect licensing? The good news is that Microsoft is licensing SQL Server by the CPU socket, not the number of CPU cores. This means that a dual CPU server running quad-core CPUs will function almost as if the server had eight CPUs, but you're only paying for two CPUs' worth of SQL Server licensing.**

Although it's limited to four CPUs with multi-core CPUs, it is a lesser limitation than in the past and there's no limit to memory. With a well-designed four-way server running multi-core CPUs and plenty of RAM, Standard Edition can easily handle 500 concurrent users and a database pushing a terabyte of data.

- **Workgroup Edition:** Intended as a departmental database server, Workgroup Edition includes the right mix of features for a small transactional database that's connected to a Standard or Enterprise Edition.

  The key feature missing from Workgroup Edition is Integration Services, the rationale being that a workgroup database is likely the source of data that is integrated by other larger database servers but does not pull data from other sources itself. This may be the single factor that drives you to move up to Standard Edition. In my experience, a database of this size does occasionally require moving data even if for an upgrade of an Access database.

  I recommend Workgroup Edition for small businesses or departments that don't require extremely high availability or Analysis Services. A server with two dual-core CPUs, 3GB of RAM, and a well-designed disk subsystem could easily serve 100 busy users with Workgroup Edition.

- **Web Edition:** As the name implies, the Web Edition, new for SQL Server 2008 is licensed for hosting websites.**SQL Server Express Edition:** This free (no upfront cost, no royalties, no redistributable fees) version of SQL S**erver is not simply a plug-**in replacement for the Access Jet database engine. It's a full version of the SQL Server Database Engine intended to serve as an embedded database within an application. Express does have some limitations: a maximum database size limit of 4GB, only one CPU socket, and 1GB of RAM.

  I'd recommend SQL Server Express Edition for any small .NET application that needs a real database. It's more than suitable for applications with up to 25 users and less than 4GB of data.

- **SQL Server Compact Edition:** CE is technically a different Database Engine with a different feature set and a different set of commands. Its small footprint of only 1MB of RAM means that it can actually run well on a mobile smart device. Even though it runs on a handheld computer, it's a true ACID-compliant Database Engine. This book does not cover SQL Server CE.

Best Practice

I've spoken out and blogged about my pro-64-bit view. Any new server will sport shiny, new 64-bit multi-core CPUs, and considering the smoother memory addressing and performance gains, I see no reason why SQL Server is not a 64-bit only product — except for, perhaps, Developer and Express Editions. Do your shop a favor and don't even consider 32-bit SQL Server for production.

- **SQL Data Services (SDS):** The database side of Microsoft Azure is a full-featured relational SQL Server in the cloud that provides an incredible level of high availability and scalable performance without any capital expenses or software licenses at a very reasonable cost. I'm a huge fan of SDS and I host my ISV software business on SDS.

  Version 1 does have a few limitations: 10GB per database, no heaps (I can live with that), no access to the file system or other SQL Servers (distributed queries, etc.), and you're limited to SQL Server logins.

CROSS-REF **Besides the general descriptions here, Appendix A includes a chart detailing the differences between the multiple editions.**

Exploring the Metadata

When SQL Server is initially installed, it already contains several system objects. In addition, every new user database contains several system objects, including tables, views, stored procedures, and functions.

Within Management Studio's Object Explorer, the system databases appear under the Databases ➪ System Databases node.

System databases

SQL Server uses five system databases to store system information, track operations, and provide a temporary work area. In addition, the `model` database is a template for new user databases. These five system databases are as follows:

- **master:** Contains information about the server's databases. In addition, objects in `master` are available to other databases. For example, stored procedures in `master` may be called from a user database.
- **msdb:** Maintains lists of activities, such as backups and jobs, and tracks which database backup goes with which user database
- **model:** The template database from which new databases are created. Any object placed in the `model` database will be copied into any new database.
- **tempdb:** Used for ad hoc tables by all users, batches, stored procedures (including Microsoft stored procedures), and the SQL Server engine itself. If SQL Server needs to create temporary heaps or lists during query execution, it creates them in `tempdb`. `tempdb` is dropped and recreated when SQL Server is restarted.
- **resource:** This hidden database, added in SQL Server 2005, contains information that was previously in the master database and was split out from the master database to make service pack upgrades easier to install.

Metadata views

Metadata is data about data. One of Codd's original rules for relational databases is that information about the database schema must be stored in the database using tables, rows, and columns, just like user data. It is this data about the data that makes it easy to write code to navigate and explore the database schema and configuration. SQL Server has several types of metadata:

- **Catalog views:** Provide information about static metadata — such as tables, security, and server configuration

- **Dynamic management views (DMVs) and functions:** Yield powerful insight into the current state of the server and provide data about things such as memory, threads, stored procedures in cache, and connections
- **System functions and global variables:** Provide data about the current state of the server, the database, and connections for use in scalar expressions
- **Compatibility views:** Serve as backward compatibility views to simulate the system tables from previous versions of SQL Server 2000 and earlier. Note that compatibility views are deprecated, meaning they'll disappear in SQL Server 11, the next version of SQL Server.
- **Information schema views:** The ANSI SQL-92 standard nonproprietary views used to examine the schema of any database product. Portability as a database design goal is a lost cause, and these views are of little practical use for any DBA or database developer who exploits the features of SQL Server. Note that they have been updated for SQL Server 2008, so if you used these in the past they may need to be tweaked.

These metadata views are all listed in Management Studio's Object Explorer under the Database ➪ Views ➪ System Views node, or under the Database ➪ Programmability ➪ Functions ➪ Metadata Function node.

What's New?

There are about 50 new features in SQL Server 2008, as you'll discover in the What's New in 2008 sidebars in many chapters. Everyone loves lists (and so do I), so here's my list highlighting the best of what's new in SQL Server 2008.

Paul's top-ten new features in SQL Server 2008:

10. **PowerShell —** The new Windows scripting language has been integrated into SQL Server. If you are a DBA willing to learn PowerShell, this technology has the potential to radically change how you do your daily jobs.
9. **New data types** — Specifically, I'm more excited about Date, Time, and DateTime2 than Spatial and HierarchyID.
8. **Tablix** — Reporting Services gains the Tablix and Dundas controls, and loses that IIS requirement.
7. **Query processing optimizations** — The new star joins provide incredible out-of-the-box performance gains for some types of queries. Also, although partitioned tables were introduced in SQL Server 2005, the query execution plan performance improvements and new UI for partitioned tables in SQL Server 2008 will increase their adoption rate.
6. **Filtered indexes** — The ability to create a small targeted nonclustered index over a very large table is the perfect logical extension of indexing, and I predict it will be one of the most popular new features.
5. **Management Data Warehouse** — A new consistent method of gathering performance data for further analysis by Performance Studio or custom reports and third parties lays the foundation for more good stuff in the future.
4. **Data compression** — The ability to trade CPU cycles for reduced IO can significantly improve the scalability of some enterprise databases. I believe this is the sleeper feature that will be the compelling reason for many shops to upgrade to SQL Server 2008 Enterprise Edition.

The third top new feature is Management Studio's many enhancements. Even though it's just a tool and doesn't affect the performance of the engine, it will help database developers and DBAs be more productive and it lends a more enjoyable experience to every job role working with SQL Server:

3. **Management Studio** — The primary UI is supercharged with multi-server queries and configuration servers, IntelliSense, T-SQL debugger, customizable Query Editor tabs, Error list in Query Editor, easily exported data from Query Results, launch profiler from Query Editor, Object Search, a vastly improved Object Explorer Details page, a new Activity Monitor, improved ways to work with query plans, and it's faster.

And for the final top two new features, one for developers and one for DBAs:

2. `Merge` and **Table-valued parameters** — Wow! It's great to see new T-SQL features on the list. Table-valued parameters alone is the compelling reason I upgraded my Nordic software to SQL Server 2008. Table-valued parameters revolutionize the way application transactions communicate with the database, which earns it the top SQL Server 2008 database developer feature and number two in this list.

   The new `merge` command combines insert, update, and delete into a single transaction and is a slick way to code an upsert operation. I've recoded many of my upsert stored procedures to use `merge` with excellent results.

1. **Policy-based management (PBM)** — PBM means that servers and databases can be declaratively managed by applying and enforcing consistent policies, instead of running ad hoc scripts. This feature has the potential to radically change how enterprise DBAs do their daily jobs, which is why it earns the number one spot on my list of top ten SQL Server 2008 features.

Going, Going, Gone?

With every new version of SQL Server, some features change or are removed because they no longer make sense with the newer feature set. *Discontinued* means a feature used to work in a previous SQL Server version but no longer appears in SQL Server 2008.

Deprecated means the feature still works in SQL Server 2008, but it's going to be removed in a future version. There are two levels of deprecation; Microsoft releases both a list of the features that will be gone in the next version, and a list of the features that will be gone in some future version but will still work in the next version.

Books Online has details about all three lists (just search for deprecated), but here are the highlights:

Going Eventually (Deprecated)

These features are deprecated from a future version of SQL Server. You should try to remove these from your code:

- SQLOLEDB
- `Timestamp` (although the synonym `rowversion` continues to be supported)

continued

continued

- `Text`, `ntext`, and `image` data types
- Older full-text catalog commands
- `Sp_configure 'user instances enabled'`
- `Sp_lock`
- SQL-DMO
- Sp stored procedures for security, e.g., `sp_adduser`
- Setuser (use `Execute as` instead)
- System tables
- `Group by all`

Going Soon (Deprecated)

The following features are deprecated from the next version of SQL Server. You should definitely remove these commands from your code:

- Older backup and restore options
- SQL Server 2000 compatibility level
- `DATABASEPROPERTY` command
- `sp_dboption`
- `FastFirstRow` query hint (use `Option(Fast n)`)
- ANSI-89 (legacy) outer join syntax (*=, =*); use ANSI-92 syntax instead
- `Raiserror` integer string format
- Client connectivity using DB-Lib and Embedded SQL for C

Gone (Discontinued)

The following features are discontinued in SQL Server 2008:

- SQL Server 6, 6.5, and 7 compatibility levels
- Surface Area Configuration Tool (unfortunately)
- Notification Services
- `Dump` and `Load` commands (use `Backup` and `Restore`)
- `Backup log with No_Log`

continued

continued

- `Backup log with truncate_only`
- `Backup transaction`
- `DBCC` Concurrencyviolation
- `sp_addgroup`, `sp_changegroup`, `sp_dropgroup`, and `sp_helpgroup` (use security roles instead)

The very useful Profiler trace feature can report the use of any deprecated features.

Summary

If SQL Server 2005 was the "kitchen sink" version of SQL Server, then SQL Server 2008 is the version that focuses squarely on managing the enterprise database.

Some have written that SQL Server 2008 is the second step of a two-step release. In the same way that SQL Server 2000 was part two to SQL Server 7, the theory is that SQL Server 2008 is part two to SQL Server 2005.

At first glance this makes sense, because SQL Server 2008 is an evolution of the SQL Server 2005 engine, in the same way that SQL Server 2000 was built on the SQL Server 7 engine. However, as I became intimate with SQL Server 2008, I changed my mind.

Consider the significant new technologies in SQL Server 2008: policy-based management, Performance Data Warehouse, PowerShell, data compression, and Resource Governor. None of these technologies existed in SQL Server 2005.

In addition, think of the killer technologies introduced in SQL Server 2005 that are being extended in SQL Server 2008. The most talked about new technology in SQL Server 2005 was CLR. Hear much about CLR in SQL Server 2008? Nope. Service Broker has some minor enhancements. Two SQL Server 2005 new technologies, HTTP endpoints and Notification Services, are actually discontinued in SQL Server 2008. Hmmm, I guess they should have been on the SQL Server 2005 deprecation list.

No, SQL Server 2008 is more than a SQL Server 2005 sequel. SQL Server 2008 is a fresh new vision for SQL Server. SQL Server 2008 is the first punch of a two-punch setup focused squarely at managing the enterprise-level database. SQL Server 2008 is a down payment on the big gains coming in SQL Server 11.

I'm convinced that the SQL Server Product Managers nailed it and that SQL Server 2008 is the best direction possible for SQL Server. There's no Kool-Aid here — it's all way cool.