CHAPTER 3

Enterprise Data Management

In the beta history of the SQL Server 2005 release, significant emphasis has been put on the new features for developing applications. Let's just say that Microsoft loves developers. But what happened to the features for those who get paged in the middle of the night when mission-critical systems hang? This chapter looks at features for the unsung heroes of corporate IT—the database administrator (DBA). For the DBA, SQL Server 2005 will change everything about the way administration is accomplished.

This chapter covers what's new in enterprise database administration. It doesn't just look at the product from a feature list, but groups the features according to some tasks common to database administration. This chapter specifically discusses the following:

- **Infrastructure management.** How do installation and configuration work?
- **SQL Server monitoring.** We'll divide this into reactive and proactive and see how it gets done.

Then we'll look at one of the major efforts for the SQL Server 2005 product—the emergence of Very Large Database (VLDB) management—especially in the area of backup and recovery. Along the way, we'll cover replication, high availability, general data availability, and business intelligence for the database admin.

Before we get to those features, though, we should look at the new tool set, because it is a major shift from Microsoft Management Console–(MMC) based tools to Visual Studio. For the database developer and administrator, user interaction and the tool set have many similarities. Hopefully, in the end, this will lead to greater productivity and better-quality database applications.

At a Glance: What's New for the Database Administrator

Microsoft SQL Server 2005 represents the cohesion of the developer and administrator instrumentation. Of all the Microsoft Server products, SQL Server has made the most significant progress related to scalable and extensible database management and authoring tools. SQL Server management tools now include the following:

- A new authoring, management, and operations tool suite. Enterprise Manager, Query Analyzer, and more have been replaced with an integrated tool set known as SQL Server Management Studio.
- New APIs are included for remote management of SQL Server database servers.
- New technologies are introduced for removing barriers to availability, both general and high availability, via a portfolio of technologies.
- Routing administrative tasks such as backup and restore have been enhanced to decrease the maintenance and recovery windows, allowing for greater database availability.

The new management tools range from the small, such as creating synonyms for database objects, to the dramatic, such as the introduction of .NET assemblies into the database. What's more, the tools used by DBAs have been completely redesigned and rewritten.

When we look at SQL Server 2005 from a database administration point of view, we can group the features around certain job functions. For example, what features does SQL Server 2005 have for remote management of servers, including setup? What features will allow a DBA to find and mend a blocking process or a poorly configured stored procedure? One of the big challenges that DBAs face is how to keep database systems available as applications and databases constantly change. What about the mundane but important tasks, such as disaster planning, security maintenance, resource allocation, and modeling of future resource needs? How does SQL Server 2005 address these issues?

Additionally, as database products conform more to standards, and the manufacturers copy each other, the question then becomes, "What's innovative?" What has Microsoft delivered in features that will solve a problem you will have tomorrow because Microsoft is thinking ahead? When we look at SQL Server 2005, it's important to separate the features from the marketing message. Sure, Database Snapshot is new and innovative, but is it useful? The release of any product is a combination of reaching for future capabilities—the next big thing—and making the product solve the most common issues, making the product more complete. In the case of Database Snapshot, its usefulness is constrained by its usage scenarios. It's always interesting to hear from customers how they found a new use for a particular technology. Sometimes, these creative usages are the cause of customers' issues. Other times they find legitimate new uses that then influence new features in the next release. This chapter looks at the database management features not as a feature list, but from a DBA task orientation perspective. Before we do that, however, we must look at the center of all the change: SQL Server Management Studio.

SQL Server Management Studio

In SQL Server 7 and 2000, the tools suite was based entirely on MMC. The MMC tool is not designed for real scalability. Customers complained that Enterprise Manager took a long time to open extremely large databases with complex schema. The fact is that Microsoft develops products in a "We'll get there" style. SQL Server Management Studio is one of the few "We've gotten there" tools supplied by Microsoft. Compared to other management tools delivered by Microsoft, SQL Server Management Studio is brilliant. In contrast, you can look at replication; it's still lacking in clear tools strategy. First, know that SQL Server Management Studio is built on the same underpinnings as Visual Studio 2005. Things such as Help and the myriad of panes can really clutter up your screen. On the upside, SQL Server Management Studio includes the following:

- Full support for management of instances of SQL Server 7.0, SQL Server 2000, SQL Server 2005, and Analysis Services 2005. Management Studio dialogs automatically customize to show only the appropriate choices and features, depending on the version of the database server the user is working with. Nonmodal dialogs allow the user to multitask and do more things at once.
- A new integrated Query Editor lets you create queries for all the SQL Server technologies. Additionally, the Query Editor has customization capabilities that make it easier to work with large batch files and complex queries.

- Built-in support for source control. Whether you're using Microsoft SourceSafe or Visual Studio Team System, SQL Server files can be controlled in the same way as other development pieces. You can use any source-control system that uses the Source Safe Control Interface API.

The SQL Server Management Studio implements the SQL Server Management Objects (SMO), which is a new set of managed classes that replace the SQL Server Data Management Objects (SQL-DMO). This major architectural change brings significant enhancements in performance.

SMO's first important optimization over SQL-DMO is delayed instantiation. As you run your application, SMO retrieves objects and properties as needed. You'll notice this right away in the Object Explorer. The key to this optimization is making many small round-trips to the server instead of getting everything up front, as SQL-DMO does and which is overkill in many scenarios. SMO also lets you prefetch entire collections. In addition, you can retrieve objects by using a set of predefined properties. The bottom line is that the programmer has control over SMO behavior, which lets you build an application that suits your needs.

The SMO object model is also cached, meaning that it doesn't propagate object changes to the server immediately. Instead, it caches them until you decide to apply (or discard) the changes. This caching yields fewer round-trips to the server because all changes are sent as one set of batches.

SMO provides advanced scripting functionality as part of the new Scripter object. This object lets you discover database-object dependencies, which results in an object tree. You can create an ordered object list from that tree and then generate a script from the list and optionally specify scripting options (a superset of SQL-DMO's scripting options). This architecture gives you maximum control over each scripting phase, letting you build specialized, customized scripting solutions.

Additionally, SMO includes a script-capture mode that lets you capture the Transact-SQL code that SMO generates when your application performs an operation on an object. For example, a Visual Basic guru can use SMO to grab the Transact-SQL that his or her application generates.

Now that you understand the architectural structure of the how the Management Studio works, let's look at the tools in more detail.

A Connected or Disconnected State

Before you get started with SQL Server Management Studio, you'll notice something radical. The old Enterprise Manager user interface is gone. Moreover, you now have a new connection dialog to work with. The connection dialog allows users to provide both logon credentials and specific connection properties. The connection dialog can connect directly to SQL Server Engine, Analysis Services, Reporting Services, Integration Services, and SQL Server Mobile Edition. The Mobile Edition connection is interesting because the mobile database is often found on a Pocket PC device.

Besides being able to connect to previous versions—meaning SQL Server 2000 and SQL Server 7.0—the connection dialog allows you to decide which database, network method (TCP/IP), named pipes, and shared memory is used to connect. Additionally, you can choose to encrypt your connection and provide specifics such as database, connection time-out, and network packet size.

As soon as you are connected to a database server, you immediately notice the new layout of the windows. As with previous versions, you can view registered servers. You can review the database objects found under the registered server via the new Object Explorer window. It's important to note that you see only objects you have security permissions for.

Object Explorer

At the highest level, the database, not the server, is the central container for all the objects found therein. This is a significant departure that makes good sense. First, it's more secure, because the metadata security found in SQL Server 2005 allows for least privileges—all the way down to the database. On another level, having the database as the central axis for all the objects related to the database makes administration easier. When you click the plus symbol next to a database, only those objects directly related are sent back. In previous versions, you had to write queries to get back all the objects related to a database. In large database deployments such as SAP, this new organizational model is a time-saver.

Microsoft strives to not do any take-backs on features between releases. The new SQL Server Management Studio is no exception. I recommend taking the time to get to know the features found in SQL

Server 2000 that are renamed and moved in SQL Server 2005. You can still do the following:

- Create a database diagram
- Create database tables via the Visual Database Tools (VDT)
- Create security objects
- Create replication objects such as publications
- Monitor replication

To create a database diagram, you will find a new folder called Database Diagrams under the specific database objects found in the Object Explorer. In Figure 3-1, notice how the database is now the new container for all the subordinate objects. This is a big improvement over previous versions.

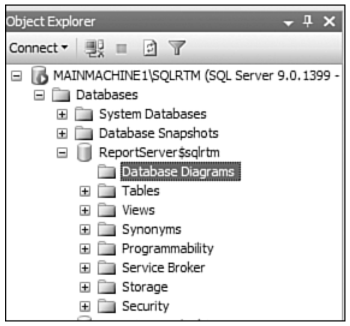


Figure 3-1 Object Explorer.

Creating a Database

There are multiple ways to create a database in SQL Server 2005. My favorite is to right-click the Databases folder in the Object Explorer and select New Database. This pulls up the friendly new Create Database dialog. This nonmodal dialog is an easy-to-use tool for getting all your database settings worked out. It has three tabs. The first, General, supplies the needed naming text boxes. The second tab, Options, contains more specific settings such as auto-shrink and auto-close. You can also specify the cursor behavior, several miscellaneous settings (as Microsoft calls them), and recovery and state values. The third tab, Filegroups, is where you build file group allocations.

The Create Database dialog gives you the option of scripting all the settings to a Query Editor window, a file, the Clipboard, or a job. I find this new feature quite handy. I usually script my new databases, save the files in SourceSafe, and then start to build the objects. If your company has a standard database format, the scripting feature will not be lost on you.

Creating Tables

With your database in place, you can start creating tables. If you are designing a table structure from scratch, you can use the VDT and create the database via a database diagram, or you can right-click the table folder and select New Table. I prefer to work with the database diagram, because I like to build relationships between tables visually. Whenever you save the database diagram, the tables are created. This makes iteration very easy. Generally, I try to create all my tables, right-click the database icon, and select Generate Scripts to capture all the changes to the database structure. The script is again checked into source control to allow for rolling back if there's an issue.

If you aren't a visual person, you might want to use the built-in templates. You can find them in the new Template Explorer box; select View, Templates. With your database and tables in place, let's look at the new Query Editor.

Query Editor

SQL Server Management Studio contains a host of new features. One of the first tools you'll use is the new Query Editor (QE), which replaces Query Analyzer. QE is much more than a simple query text writing application. QE provides the following:

- Disconnected editing to allow access to the Query Editor without establishing a connection to an instance of SQL Server.
- Color coding of Transact-SQL syntax to improve the readability of complex statements.
- Automatic statement formatting, including automatic indenting.
- Templates that can be used to speed development of Transact-SQL statements for creating SQL Server objects. Templates are files that include the basic structure of the Transact-SQL statements needed to create objects in a database.

- Editing of execute and parse queries with Object Linking and Embedding (OLE) SQL keywords.
- Support for query editing on multiple versions of SQL Server, including SQL Server 7.0, SQL Server 2000, and SQL Server 2005.
- Results presented in either a grid or a free-form text window.
- A graphical diagram of the showplan information showing the logical steps built into the execution plan of a Transact-SQL statement.
- The ability to organize work items into solutions, projects, and files using a specialized folder structure.

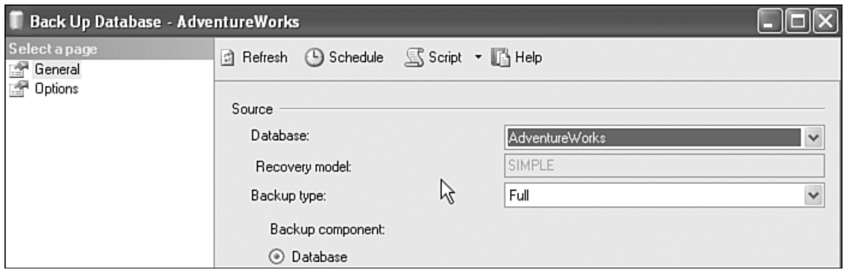
Nonmodal Dialogs

One of the more interesting and useful changes to how DBAs and developers will use SQL Server Management Studio is found in the new user interface dialog boxes. In previous versions of SQL Server, an administrator would use the Backup Database Wizard and execute a backup job. The dialog for accomplishing this was modal, meaning that you had to wait until the job finished. In very large or slow operations, this wastes considerable time. SQL Server 2005 changes this with nonmodal dialogs. The new dialog box style provides more of the information you need to accomplish a certain task, but the walk-through wizard is gone.

Let's say a DBA needs to perform several tasks, including creating a backup job, writing a Transact-SQL script to create a database, and adding a user account. The administrator launches SQL Server Management Studio and creates and executes the backup job. The backup job takes some time to complete, but the administrator can perform the other tasks because the backup dialog is no longer modal. The new dialog style offers the following features:

- Scripting from any dialog box. Administrators and developers can create a script from any dialog box so that you can read, modify, store, and reuse the scripts. Scripts can be written directly to a Query Editor window, to a file, or to the clipboard.
- Scheduling or immediate execution of management actions. Every management action can be scheduled in the SQL Server Agent or run immediately.

As you can see in Figure 3-2, the nonmodal dialog is both more complex and more flexible. Managers shouldn't hear database administrators say they are waiting on an action to complete before doing the next task.



Source
Database: AdventureWorks
Recovery model: SIMPLE
Backup type: Full
Backup component:
Database

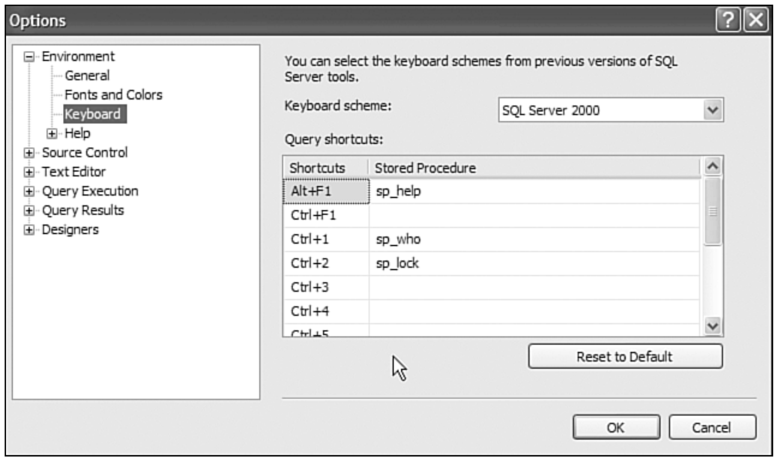
Figure 3-2 Nonmodal dialog box.

Now that you understand the basics of SQL Server Management Studio, we can look at how SQL Server 2005 changes how day-to-day tasks are accomplished. One last thing to remember about SQL Server Management Studio is that you can only see and act on items at the level of security authorization for the login used to connect to SQL Server.

Customizing the SQL Server Management Studio

After you get over the shock of the new layout, you'll want to start changing it to meet your work style. You have several ways to customize the look and feel of the "shell." You can use the Views menu to add and remove toolbars. If you are familiar with Visual Studio, these are easy to understand. Additionally, you can change the look and feel of the free-form Query Editor by using line numbering. When you use the Go verb, you can collapse and expand large blocks of text.

One of the more useful changes you can make is changing the keyboard scheme to reflect SQL Server 2000. By default, the F5 key doesn't execute the current window queries. Figure 3-3 shows the Options window, with the keyboard scheme set to SQL Server 2000. You can customize any of the keys, which can save typing and possible headaches.



You can select the keyboard schemes from previous versions of SQL Server tools.

Keyboard scheme: SQL Server 2000

Query shortcuts:

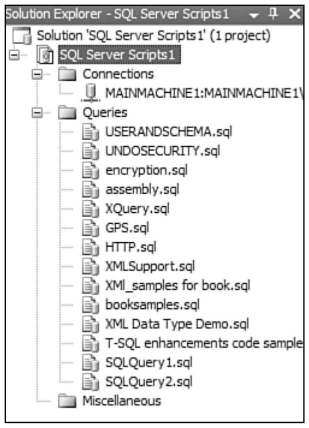
Shortcuts	Stored Procedure
Alt+F1	sp_help
Ctrl+F1	
Ctrl+1	sp_who
Ctrl+2	sp_lock
Ctrl+3	
Ctrl+4	

Figure 3-3 The Options screen, found under the Tools menu in SQL Management Studio.

Projects and Solutions Using SQL Server Management Studio

When Microsoft decided to bring together the database management and development tool set, one of the key attractions was the ability to use a source control system to manage database projects. SQL Server Management Studio can take advantage of any source control system. More importantly, SQL projects can now be organized via a project hierarchy. To use a project, simply select File, New Project in Management Studio. Figure 3-4 shows a typical SQL Server project.

It's a little confusing how Microsoft has set up this system. When you go to the File menu to create a new project, you see the solution name included in the dialog. The folders are organized with one automatically created project. The project contains folders for connections, queries, and miscellaneous items. If you right-click the solution name, which is the highest-level folder, you can add and remove projects and even import other projects. This method of organization allows for easier working. I find projects useful, because you can have a single solution with projects that contain items for each phase of development. Combining this methodology with source control means that I have an organized and efficient approach to working with database objects.



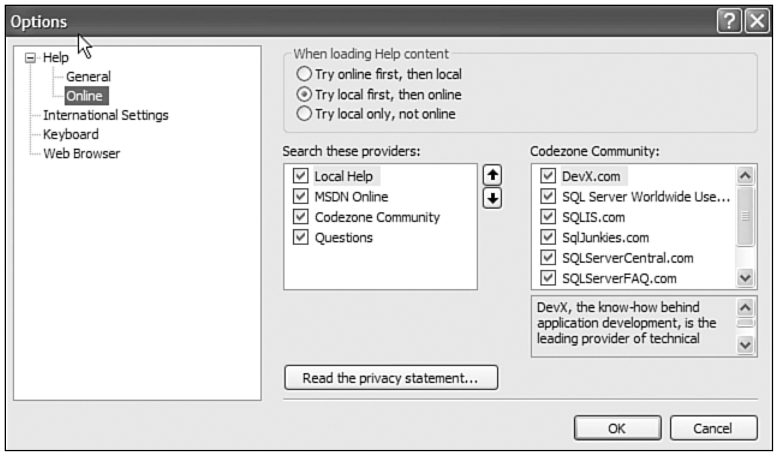
```
Solution Explorer - SQL Server Scripts1
Solution 'SQL Server Scripts1' (1 project)
  SQL Server Scripts1
    Connections
      MAINMACHINE1:MAINMACHINE1\
    Queries
      USERANDSCHEMA.sql
      UNDOSECURITY.sql
      encryption.sql
      assembly.sql
      XQuery.sql
      GPS.sql
      HTTP.sql
      XMLSupport.sql
      XMl_samples for book.sql
      booksamples.sql
      XML Data Type Demo.sql
      T-SQL enhancements code sample
      SQLQuery1.sql
      SQLQuery2.sql
    Miscellaneous
```

Figure 3-4 Project folder hierarchy.

The ability to define a connection, or a connected use for each object in my project, has some usefulness. Let's say that you're developing an application that has several users. Each user has a specific set of privileges, and those privileges affect query execution. You could create a separate connection for each user. With this connection and associated query, you could test the query under the user's security roles, which helps expose issues with the batches and security settings.

Getting Help

Although it doesn't need an entire chapter, the new SQL Server Books Online contains some new functionality worth mentioning. Books Online now includes not only local search capability, but also configurable automatic/simultaneous searching on the Internet. The Books Online Internet search gives you results from MSDN, CodeZone Communities, and more. You can customize the search results. Open Books Online by clicking the F1 button, and select Tools, Options. Figure 3-5 shows the options available for searching.



When loading Help content
- Try online first, then local
- [✓ Try local first, then online]
- Try local only, not online

Search these providers:
- ✓ Local Help
- ✓ MSDN Online
- ✓ Codezone Community
- ✓ Questions

Codezone Community:
- ✓ DevX.com
- ✓ SQL Server Worldwide Use...
- ✓ SQLIS.com
- ✓ SqlJunkies.com
- ✓ SQLServerCentral.com
- ✓ SQLServerFAQ.com

DevX, the know-how behind application development, is the leading provider of technical

Figure 3-5 Help online search options.

SQL Server 2005 Infrastructure Management

One of the largest consumers of a database administrator's time is working across the infrastructure to ensure that databases are available for applications while service packs are deployed, application code and underlying database schemas are changed, and the data itself is changing. Add to this the complexity of systems, a lack of transparency or complete knowledge of the database server infrastructure, and the fact that the database may be in another country, and you can see what a challenge managing change is for the DBA. The goal for you, the DBA, is to keep the plane in the air while the engines are being changed.

Microsoft has made some progress in the area of change management. One of the most potentially painful topics is setup, either local or remote. It seems that as fast as hardware models change, operating system and application code changes to keep pace. In organizations with high data growth rates, the underlying hardware structures are consumed and become potential bottlenecks. Ultimately, managing change is a combination of administrative skills, business process, and product features. Combined, these constructs reflect the organization's overall data management strategy. Change management has been one of the biggest issues for SQL Server 2000. SQL Server 2005 has improved change management scenarios in several key areas:

- Database Snapshot lets you roll back a change in a database schema or data by reapplying changed database pages.
- Setup. Using an MSI-based model, local and remote installations can be accomplished. Setup now provides a setup consistency checker that examines and provides reporting about the receiving server's state. Setup can also be executed from the command line and is fully scriptable.
- SQL Server Management Studio includes support for source control. This allows for tighter control of queries and batch files in the system. Source control is extended to SQL Server Integration Services (SSIS) and business intelligence database code.
- Remote management and scriptability. SQL Server 2005 supplies several methods to script all objects in the database. These scripts can be used to manage database code via source-safe technologies. Additionally, these scripts can be used to generate database objects, including OLAP databases in remote locations.
- For testing and development phases, the new Dynamic Management Views increase visibility into memory and system processes, allowing for query and procedure evaluation.

Maintaining availability during state changes on the server infrastructure happens at several levels. At the hardware level, SQL Server conforms to capabilities provided by Windows 2003 and the hardware original equipment manufacturer (OEM). SQL Server takes advantage of dynamic capabilities for adding and removing RAM, disc arrays, and components without restarting the server from Windows Server 2003. With Windows 2003, many server changes still require restarting the server. By understanding the Windows environment, administrators can be prepared to deal with planning downtime for maintenance. Their strategy should be to employ new features of SQL Server 2005 to mitigate planned downtime.

Two new features help in this capacity—database mirroring and Database Snapshot, which you can think of as a mechanism for rolling back changes to database objects. Database Snapshot has several advantages that make it perfect for dynamic system changes. Database Snapshot uses a technology called copy-on-write. This mechanism captures in the Snapshot Database only data pages that have changed. This makes the snapshot very lightweight. A development team could take a snapshot before applying any schema changes.

Another tool for managing infrastructure change is the Setup program. Now in SQL Server, an administrator can make changes to database features, including adding them to and removing them from failover clustering using the Setup dialogs. (Setup is discussed in depth in the section "What's New in Setup?")

Database Snapshot

Database Snapshot is a read-only copy of the database. Database Snapshot isn't meant to be used for reporting or as a snapshot of the database from which to develop new schema. It is actually a sparse file with pointers to the original data pages. Only when the pages change does the snapshot absorb the original page. At creation time, Database Snapshot creates a sparse file and bitmap. They are stored in memory in the buffer cache memory allocation. The size of the bitmap is directly related to the size of the source database. This is an important consideration on smaller RAM systems. In a system with a total of 4GB of RAM, if the database is large, the memory pool could have pressure applied to it via too many snapshots residing in the buffer pool. It makes sense to implement a policy that all changes to the server, except RAM and hard drives, should be accomplished after a Database Snapshot is created. It's also a good practice to delete any unneeded Database Snapshots.

With their simplicity of use, snapshots are easily abused, so you should use them carefully. The files need understandable names. If you generate a lot of them, consider a naming convention like this:

*Databasename_datatype_year_day_militaryhour_*.sht

The file extension can be almost anything. Books Online uses SS. I use sht. You can see how easy it is to create a snapshot in the following sample:

```
CREATE DATABASE AdventureWorks_dbss1800 ON
( NAME = AdventureWorks_Data, FILENAME =
'C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Data\AdventureWorks_data_1800.sht' )
AS SNAPSHOT OF AdventureWorks;
GO
```

What's New in Setup?

In SQL Server 2005, Setup has dramatically improved over previous versions. This is due largely to the move from a third-party installer to Microsoft Windows Installer. The core Windows Installer technology resides on all Windows operating system installations and provides an excellent backbone for installing SQL Server. SQL Server Setup has two installation modes:

- Unattended Setup is used for remote installations. It includes the ability to set up failover clustering on a remote server.
- Attended Setup is done interactively through a wizard. The new wizard takes users through each step of the installation process. One of the most important gains from the new Setup tools is a higher level of security. The Setup Wizard ensures that all the features are installed securely by default.

Windows Installer

The Windows Installer installs all components in a single feature tree. Minimum and typical installation modes are no longer implemented. Instead, Setup displays a feature tree with default options selected. Administrators can then customize the installation by selecting and clearing items on the feature tree and specifying installation paths. This version of Windows Installer also supports remote Setup and multiple instance configurations.

SQL Server 2005 uses Add or Remove Programs in Control Panel to add or remove individual features and to remove instances of SQL Server. Maintenance of existing SQL Server instances is supported via the Setup user interface, the command line, Microsoft Systems Management Server, or with an .ini file.

SQL Server and its supporting components—Analysis Services, Reporting Services, and Notification Services—are now integrated into a single feature tree. Setup for SQL Server 2005 provides advanced detection logic to identify previous component installations, simplifying installation of additional components or instances and upgrade of existing instances of SQL Server. One of the most exciting new features of Setup is the Consistency Checker.

Setup Consistency Checker

SQL Server 2005 provides Setup Consistency Checker (SCC), a new feature that checks and validates the target computer before Setup begins. Using Windows Management Instrumentation (WMI) technology, SCC prevents Setup failures due to unsupported configurations on local, remote, and clustered target computers. If Setup can fix failed check items, the user can allow Setup to take the necessary action. Otherwise, SCC guides the user to a solution for each blocking issue before Setup can continue. SCC provides a rich reporting interface that allows you to click through to help files. Additionally, the report can be copied to the clipboard or e-mailed from the Report window. SCC's profile changes according to the feature selected for installation.

Customizable Installation Path

Setup allows administrators to specify custom installation paths for the main features of SQL Server, including Analysis Services, Reporting Services, SQL Server Relational Database, and Client Tools. In addition, administrators can customize install paths for SQL Server log files and tempDB. The ability to customize installation paths is useful in the following scenarios:

- An administrator wants to install tempDB and the log files to different volumes on a file system. During setup, the administrator can configure custom installation paths for tempDB and the log files.
- An administrator wants to install Analysis Services to a different location than the default defined in Setup. The administrator can configure a custom installation path for Analysis Services.

Failure Reporting

One of the biggest headaches for DBAs is when Setup fails. In previous versions of SQL Server, if the installation failed, you simply had to check the log files and hope you could figure out what happened from the cryptic log text. With SQL Server 2005, significant work has been done to make Setup as painless as possible.

Setup for SQL Server 2005 includes improved failure reporting and extensible alerts. If an error occurs during installation, Setup determines a failure exit code, provides a descriptive error message, recommends corrective actions to take for resolution, and points the user to the Setup log. Setup also saves the log from each installation.

For example, suppose that while upgrading from SQL Server 2000 to SQL Server 2005, an administrator receives a Setup error. The administrator is presented with an informative alert stating that a specific dynamic link library (DLL) failed to register on the system. The administrator clicks OK, and Setup rolls back the failed installation. When Setup is done rolling back the failed installation, the administrator sees a dialog box that asks if he or she wants to report the problem to Microsoft and is directed to any additional help available. The administrator can report the problem on the Microsoft Support website and receives a link to a related Knowledge Base article. The article presents the administrator with a possible workaround for the problem.

Watson Integration

SQL Server 2005 extends exception handling in replication components to include local minidump files for Dr. Watson 1.0 integration. In earlier versions of SQL Server, if a replication agent hit an exception, a stack dump was generated. This dump provided debug information but was not generated in a format that could be loaded in Visual Studio and debugged. In SQL Server 2005, a Visual Studio–compatible minidump file is generated.

Applications can produce user-mode minidump files that contain a useful subset of the information contained in a crash dump file. Applications can create minidump files quickly and efficiently. Because minidump files are small, they can be easily sent over the Internet to technical support for the application.

The dump file is stored in the mssql\log folder for replication executables and in the current directory for any .exe using replication ActiveX components. In addition to the local minidump, replication components also call Dr. Watson to generate a cab file with debug information.

Operations Management Tools

Of the many tasks performed by DBAs, the task of operations management is really the oversight of the SQL Server infrastructure. It is divided into client connection configuration and overseeing and managing batch processes or jobs that run via the SQL Server Agent. Included in this is the management of replication and SQL Server Integration Services, as well as Reporting Services and Notification Services. Each of these technologies can be started and stopped from SQL Server Management Studio. From a pure management perspective, Microsoft has managed to integrate the current feature set more uniformly and consistently than in previous versions. In SQL Server 2005, the DBA can choose how operational tasks will be accomplished.

SQL Computer Manager

SQL Computer Manager allows administrators to configure basic service and network protocol options. SQL Computer Manager combines the functionality of the following SQL Server 2000 tools: Server Network Utility, Client Network Utility, and Service Manager. SQL Computer Manager is an MMC snap-in, similar to Enterprise Manager in previous versions. SQL Computer Manager also includes the ability to set service properties for the following services:

- SQL Server
- SQL Server Agent
- Analysis Server
- Microsoft Search
- Distributed Transaction Coordinator (DTC)
- Reporting Services

SQL Computer Manager displays all services, server network protocols, and client network protocols. It allows administrators to start, stop, pause, resume, or restart a service. It also lets you view properties for a selected service, including

- Name
- Description

- Status (started, stopped, or paused)
- Startup type (manually, automatic, or disabled)
- Log On As (the service account that the service runs under)
- Last Start Date (the service's last known start date)
- Last Stop Date (the service's last known stop date)
- Process Identifier (PID)

SQL Server Agent

SQL Server Agent automates recurring jobs performed on a server. The current version has dealt with the majority of issues that SQL Agent had in SQL Server 2000. SQL Server Agent operates as a Windows Service and can be started and stopped from SQL Computer Manager. SQL Server Agent can used to monitor servers, run jobs such as backups, and monitor server conditions via alerts. One thing customers have asked for—and that is true of SQL Server 2005—is that SQL Server Agent be included in failover clustering.

One of SQL Server Agent's newest capabilities is that it can perform jobs for other SQL Server subsystems. For example, SQL Server Integration Services and Analysis Services jobs can now be automated. The combination of the new nonmodal dialog boxes and SQL Server Agent should pave the way for greater levels of automated server work. From a security viewpoint, SQL Server Agent runs under the least privileges possible to execute a job. Moreover, SQL Server Agent jobs can take advantage of the "Run As" functionality, meaning that SQL Server Agent jobs no longer need to run as database owner (DBO), which has all the power over SQL Server.

One issue that was prevalent in SQL Server 2000 was SQL Server Agent jobs hanging the server. SQL Server 2005 provides new objects and counters for SQL Server Agent that System Monitor can use to track SQL Server Agent activity, including enabled jobs, enabled alerts, enabled schedules, active jobs, active alerts, and percentage of job success. The counters also let you configure alerts that are triggered by SQL Server Agent performance counter conditions.

SQL Server 2005 Remote Management Features

One of the real challenges facing administrators is how to manage more and more servers. Operations such as manufacturing and customer support are done around the globe, with local copies of data being synchronized during smaller and smaller windows. Administrators need to be able to remotely install, monitor, troubleshoot, and maintain remote servers. The SQL Server tools team has delivered three categories of interfaces to manage remote environments:

- **SQLCMD** is for folks who are comfortable with command prompt applications. For customers migrating or currently using the command prompt languages, OSQL and ISQL are deprecated and have been replaced with SQLCMD. SQLCMD is a command-line executable. You invoke it at the command prompt by typing SQLCMD.
- **SQL Server Management Objects (SMO)** is for those who build user interface–oriented applications. DBAs who want to build custom management tools will find that SMO has replaced DMO. SMO is a new API architecture that overcomes the limitations of DMO. SMO is scalable, reliable, and flexible. SMO is significantly more robust than DMO, because it is used by SQL Server Management Studio to connect and work with SQL Server instances. Every function found in SQL Server Management Studio is made possible by SMO.
- **Windows Management Instrumentation (WMI)** allows the use of Windows scripting languages, such as VBScript, and it's more complicated than SMO or SQLCMD. WMI is powerful and provides deep hooks into the operating system, which is beyond the reach of SQLCMD and SMO. In an extremely complex infrastructure, the WMI provider may prove to be the most complete solution.

Remote functionality is also enhanced with new capabilities for scripting and working with Replication and Analysis Services (via respective .NET libraries), Replication Management Objects, and Analysis Management Objects.

SQLCMD

SQL Server 2005 introduces a new command-prompt utility. SQLCMD uses the OLE DB API to communicate with SQL Server, while the other utilities use the older ODBC or DB-Library APIs. SQLCMD supports the functionality of OSQL and ISQL but also introduces a richer set of commands that allow it to operate better in application scripts, such as Microsoft Visual Basic for Applications (VBA) scripts.

SQLCMD goes beyond simple command-line calls to an active server connection. It can be used a lot like old DOS scripts; that is, you can use simple text files to supply input variables to the CMD script. For example, suppose an administrator uses a text file to supply database server connection information to a script that automatically adds a new database to the standard backup-and-recovery model. Rather than run the wizard, the DBA simply supplies the new server information and runs the script. SQLCMD can also be used to run ad hoc queries against the server. Probably the most important functionality is that SQLCMD provides the dedicated admin connection a guaranteed connection to a server that is hung up. More importantly, the dedicated admin connection has bandwidth preallocated so that you can use the Dynamic Management Views to iteratively find the objectionable issue and kill the process without taking down the entire server.

You can develop SQLCMD scripts in SQL Server Query Editor by turning on SQLCMD mode. SQLCMD mode lets you use the Query Editor to create SQLCMD files. The advantage of this mode is that you can develop the script, test it in the Query Editor, and then deploy it. Overall, a DBA should be comfortable with SQLCMD, because the pithy execution of the dedicated administrator connection and writing a few queries can save the day when a server process goes astray.

To get started with SQLCMD, go to the command prompt and type **`SQLCMD`**. The basic switch for working with SQLCMD is -S, which identifies the server. To provide an instance, the command would be
`sqlcmd -S ComputerName\InstanceName`.

The authentication type has three switches:

- -E is the default and uses the local user.
- -U lets you specify a user, such as SA.
- -P is the password. Passwords are case-sensitive. If the -P option is not used, and the SQLCMDPASSWORD environment variable has not been set, SQLCMD prompts the user for a password. If the -P option is used at the end of the command prompt without a password, SQLCMD uses the default password (NULL).

You can provide input files and output files as XML. You need to understand a number of things when working with SQLCMD. To learn more, press F1 while the SQL Server Management Studio is open, and search Books Online for SQLCMD. Getting to know this feature will be a lifesaver.

SQL Server Management Objects

SQL Server Management Objects (SMO) is a set of objects that expose the functionality of SQL Server database and replication management and Analysis Services management. You can use SMO to automate repetitive or commonly performed SQL Server administrative tasks, such as programmatically retrieving configuration settings, creating new databases, applying Transact-SQL scripts, creating SQL Server Agent jobs, and scheduling backups.

SMO is implemented as a set of .NET assemblies. SMO has many improvements over DMO, including a .NET object model, partial instantiation, capture mode execution, delegated execution, objects in space, and integration with the .NET Framework. For database management application developers and advanced DBAs, SMO is the primary means of creating custom applications that can manipulate instances of SQL Server. With SMO's inclusion in the .NET Framework, you can develop both web-based and Windows Forms applications. SMO can also manage instances of SQL Server Express.

Windows Management Instrumentation

Microsoft SQL Server 2005 introduces a Windows Management Instrumentation (WMI) configuration provider to programmatically manage SQL Server configuration. With the WMI provider, you can write management applications or scripts that monitor, configure, and control management information about SQL Server using the standards-based, object-oriented, remote-enabled, scriptable interface provided by the WMI framework.

If a DBA needs an easy way to retrieve all the configuration settings for a given SQL Server instance and save these settings to a text file, WMI might provide the means. WMI can also provide a means for capturing the overall server configuration, which includes Windows operating system information. This may prove useful in a disaster situation. Having a backup of the entire Registry and all the subsettings would make it easier to re-create the server from scratch. The DBA should be familiar with writing administrative scripts in VBScript using the Windows Script Host (WSH). The WMI interface lets you write a script that uses the SQL Management WMI provider to retrieve a collection of properties for a given SQL Server and save those properties to a text file.

SQL Server Monitoring

Server monitoring is a challenge for SQL Server administrators. On average, SQL Server DBAs are responsible for twice as many servers as their peers using other platforms. The DBA also has a number of tasks to accomplish. How do DBAs get everything done in a day? They use the scripting languages just mentioned to automate their processes.

In an ideal world, databases would be self-monitoring. The database itself would kill a hanging process automatically. The creation of a new database would cause the automatic creation of backup jobs. The database would do what's needed to keep query performance above a certain accepted level. Unfortunately, none of the database providers are even close to providing this functionality. That's why DBAs are still needed.

There are two kinds of monitoring scenarios:

- **Reactive monitoring** deals more with the resolution of existing issues or those that crop up.
- **Proactive monitoring** is the process of looking at a current server state and making estimates and plans for changes to the underlying objects. The objective is to prevent issues, increase scalability, and maintain availability.

This section divides the scenarios into reactive and proactive monitoring. Reactive technologies are discussed more because that venue has significant changes.

Reactive Monitoring

Reactive tools have only one true purpose: to fix a problem as quickly as possible. SQL Server 2005 has a number of new tools for looking into the server to diagnose and resolve problems. Finding the root cause of a problem and fixing the issue requires that the server provide information—a sort of health report on what's happening. Often, an issue appears on a server, and the DBA doesn't have anything to use for a diagnosis. (It's like when you bring your car to your mechanic with a complaint about a funny noise. Unless it's something loud and horrible, the sound disappears the minute the mechanic gets behind the wheel. You're left with that "I'm not crazy!" feeling.) With SQL Server 2005, the new Dynamic Management Views don't just provide a way to look under the hood. They provide historical aggregation data for processes running under the hood. A DBA trying to solve a problem will have at least some basic clues to start the investigation.

Database Catalog Views

To better protect SQL Server databases, the system catalogs have been locked down. They can't be changed. Also, the system catalogs aren't universally visible. SQL Server provides a new set of SQL views that provide information about the system catalog. These views are read-only and are designed to show the metadata. The following is a list of the available catalog views. You can query the catalog views by looking up the appropriate system function and executing the query with the appropriate select statement filters. The naming convention for the catalog views is user-friendly.

Partition Function Catalog Views

Data Spaces and Fulltext Catalog Views
Common Language Runtime (CLR) Assembly Catalog Views
Scalar Types Catalog Views
Objects Catalog Views
Messages (for errors) Catalog Views
Service Broker Catalog Views
HTTP Endpoints Catalog Views

Server-Wide Configuration Catalog Views

Databases and Files Catalog Views
Schemas Catalog Views
Security Catalog Views
Database Mirroring Catalog Views
XML Schemas (XML type system) Catalog Views
Linked Servers Catalog Views
Extended Properties Catalog Views

NOTE I could spend an entire chapter discussing these catalog views, but I recommend that you read about them in SQL Server Books Online. Look under the topic "Catalog Views (Transact-SQL)."

Here's a simple of example of querying the Database Level View for principals:

```
Select type_desc,name,default_schema_name
From sys.database_principals
where type_desc like 'sql_user'
```

SQL Server Profiler

SQL Server Profiler provides a mechanism for capturing workloads on a server and replaying them. Profiler uses something called a trace file to capture and replay queries on the server. (You can learn more about the trace file in SQL Server Books Online. Reading this topic is well worth the time.) A number of improvements, including client trace and CPU-level trace elements, have been included. Profiler can sort and review stored procedures and all the executions on a server. SQL Server Profiler lets you do the following:

- Step through problem queries to find the cause of the problem.
- Find and diagnose slow-running queries.
- Capture the series of SQL statements that lead to a problem. The saved trace can then be used to replicate the problem on a test server where the problem can be diagnosed.
- Monitor the performance of SQL Server to tune workloads.

The data used by Profiler is captured in a trace file. The trace file contains elements that are recorded. It also contains information about which queries are executed, how long they are executed, and on what server. With SQL Server 2005, client connection information is now available. This added functionality provides a means for watching the events unfold in the round-trip of client-to-server execution.

NOTE SQL Server Profiler can be used with failover clustering to troubleshoot ghost failovers.

Although SQL Server Profiler does incur some overhead for capturing traces, this overhead is minor. SQL Server Profiler trace files can be saved, exported, shared, and benchmarked. The new Database Tuning Advisor can even use a trace to provide feedback for tuning databases.

Microsoft SQL Server 2005 introduces several enhancements to SQL Server Profiler:

- **Rollover trace files.** Profiler now can replay one or more collected rollover trace files continuously and in order.
- **New extensibility standard.** Profiler uses an XML-based definition. This new definition allows Profiler to more easily capture events from other types of servers and programming interfaces.
- **Profiling of Analysis Services.** Profiler now supports capturing events raised by Analysis Services. An administrator needs to identify performance problems with OLAP queries that are issued by a particular user account. The administrator configures SQL Server Profiler to capture login events and the associated server process identifier (SPID) for that session. The administrator also configures Profiler to capture all Query Begin and Query End events that have the same SPID recorded. Using Start Time, End Time, and Duration, the administrator can determine the queries' timing.
- **Save trace results as XML.** Trace results can be saved in an XML format in addition to the standard save formats of ANSI, UNICODE, and OEM.
- **Aggregate view.** Users can choose an aggregate option and select a key for aggregation. This lets users see a view that shows the column on which the aggregation was performed, along with a count of the number of rows that make up the aggregate value.
- **Correlation of trace events to performance monitor counters.** Profiler can correlate Performance Monitor counters with SQL or Analysis Services events.

Exportable Showplan and Deadlock Traces

Nothing is worse for a DBA than to have to find missing data or discover why the server is hanging on this transaction. As the DBA starts to investigate the database crime scene, the ability to forensically study a showplan, or to watch a deadlock happen on a trace reply, is critical. SQL

Server 2005 introduces enhancements to showplan and deadlock traces that give administrators additional ways to tune database servers. Many times, issues that are blamed on the database are actually due to poorly written store procedures or batch files that create contention. (I'll spare you the lecture on writing efficient apps.) The DBA is often called to the rescue in cases where a deadlock event seems to be occurring. A deadlock forces a transaction to be rolled back and fail because the underlying row is locked by a simultaneous write activity. In SQL Server 2005, new features are provided for deadlock detection:

- Deadlock occurrences collected through trace events are represented graphically. The graphical representation shows deadlock cycles or chains, providing you with a simpler and more intuitive method for analyzing deadlock occurrences than information collected from the trace flags used by earlier versions of SQL Server.
- Showplan results are saved in an XML format, which can later be loaded for graphical display in Query Editor.

The ability to save showplan results in an XML format provides a number of benefits for performance tuning. Showplans can be saved, transferred to another location, and viewed without the need for an underlying database. Administrators can use an exported showplan to help identify discrepancies between different in-house or remote databases. From a proactive monitoring perspective, administrators can collect baseline data on a server and later compare that data to servers as they grow or as performance characteristics change.

Database Alerts

SQL Server 2005 lets you automate the monitoring of system activity. Alerts can be set to respond to two types of events. First, when an error occurs, the database throws an error flag. This flag can be monitored for, and a response enacted appropriately. Second, alerts can be set to a severity level. This is a threshold-based method of responding to system states. For example, you might create an alert for a system event, such as when a wait-for process is running too long. The alerts can be designed in SQL Server Management Studio, via SQL Server Agent, or through Transact-SQL.

Dynamic Management Views

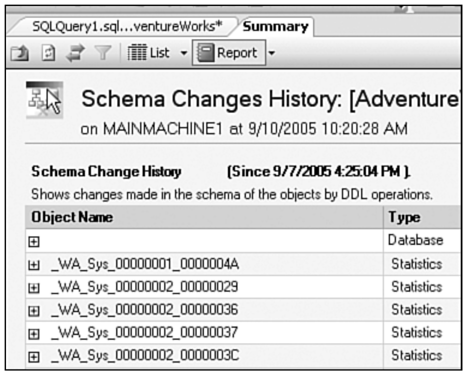
SQL Server 2005 provides more than 80 new Dynamic Management Views (DMVs). DMVs are a completely new topology for troubleshooting SQL Server database issues. They are divided into groups from the server level down to the database level. Special views are provided for checking on .NET assemblies, SQL Service Broker, security, and others. DMVs include not only current data, but also aggregated historical data.

SQL Server 2005 turns on a default trace, which provides a means for finding out what happened when an error occurs. You can think of the default trace as a black box recorder. DMVs use the default trace. This is where DMVs are really interesting: Administrators can review the default trace after an unplanned incident and see what happened. Moreover, when you first log on to SQL Server, the summary reports found in the main window are created from the historical data.

DMVs are actually database views. As such, they can be found in the Views folder under each user's Database System database folder. (The prefix for the views is dm_.) DMVs are organized into five general categories. The views are categorized by the environmental factors they report on:

- DMVs with the name dm_exec_* provide information about execution of user modules and connections.
- DMVs following the convention dm_os_* report on memory, locks, and execution scheduling.
- DMVs using the name dm_trans_* provide insight into transactions and isolation.
- DMVs for monitoring disk I/O are named dm_io_*.
- dm_db_* provides database-level data.

A DBA would be remiss not to understand these categories and build custom monitoring tools. Figure 3-6 shows the schema change report found on the Summary tab of SQL Server Management Studio. The Summary tab has 13 interesting and useful reports. Not only do these reports use the default trace and DMVs, but they actually run the Reporting Services APIs to display the data. The report shown in Figure 3-6 is interactive, allowing the user to drill into subareas of the report and probe for more information.



Schema Changes History: [Adventure

on MAINMACHINE1 at 9/10/2005 10:20:28 AM

Schema Change History **(Since 9/7/2005 4:25:04 PM).**

Shows changes made in the schema of the objects by DDL operations.

Object Name	Type
⊞	Database
⊞ _WA_Sys_00000001_0000004A	Statistics
⊞ _WA_Sys_00000002_00000029	Statistics
⊞ _WA_Sys_00000002_00000036	Statistics
⊞ _WA_Sys_00000002_00000037	Statistics
⊞ _WA_Sys_00000002_0000003C	Statistics

Figure 3-6 SQL Server Dynamic Management View report in SQL Server Management Studio.

When your server acts up and freezes, you can combine the DMVs with another new feature: the Dedicated Administrator Connection (DAC). With the DAC and DMVs, you can find the offending process or job and kill it without restarting the server.

Proactive Monitoring

In moving to a more proactive role for DBAs, organizations can prevent future issues with applications. The DBA can provide insight into hardware configurations, disk and storage needs, and performance calculations for applications. Having a seamless interface between development and administration in an organization improves overall application quality, because DBAs often are more well versed in Transact-SQL than developers and are more comfortable designing database objects. Also, the application process should take into consideration the backup/recovery strategy, should be implemented as part of the overall database management strategy, and should conform to current regulatory and privacy policy issues. Why would backup and recovery be part of an application design process?

SQL Server 2005 has two new database technologies that take advantage of specific database designs. Let's look at the partitioning feature to see how this works. The new partitioning feature works best when the database objects are designed with a partitioning scheme in mind. Moreover, the queries that peruse the data are more scalable because they more efficiently execute disk reads and file I/O. It's essential to divide table data into smaller partitions as the database grows. The use of physical file groups across possibly hundreds of disks requires specialized knowledge—which is usually the domain of the administrator. And while the application developer needs to do nothing, the database design needs to have appropriate primary and foreign key relationships that make managing the data over the long term as easy as possible. For database partitioning to work correctly, the data partitioning scheme and functionality must follow the cardinal relationships found in the data, such that moving data in and out of the partitioning system is efficient. The new partitioning and snapshot isolation functionality are covered later in this chapter.

The current breed of tools for tuning SQL Server are the work of Microsoft Research. The DMX team has provided a much-improved Database Tuning Advisor (DTA).

Database Tuning Advisor

In SQL Server 2005, the DTA replaces the Index Tuning Wizard. The DTA improves the quality of tuning recommendations, increases the scalability of tuning operations, and simplifies the user experience. The DTA includes the following improvements to database tuning:

- **Time-bound tuning.** Provides recommendations for a user-specified amount of time. In earlier versions, a user could specify only one of three tuning modes: fast, medium, or thorough.
- **Multidatabase tuning.** Tunes from workloads that reference tables in multiple databases. In earlier versions, the wizard tuned tables belonging to only one database.
- **Event tuning enhancements**. Enhanced event parsing capability allows the wizard to tune events containing table-valued user-defined functions and events referencing temporary tables.
- **Selective creation of indexes.** Users can selectively create a subset of the indexes recommended by the DTA.
- **XML output.** In addition to the Transact-SQL scripts and text analysis reports that are the standard output of the DTA, users can generate output in XML format.
- **Data partition tuning.** Recommends appropriate data partitions based on a workload.

- **Scalability enhancements.** Enhanced parsing capabilities, statistics creation, query optimization, workload compression, and memory management are designed to increase the scalability of tuning operations. The advisor also uses multiple connections to perform work in parallel on a server where multiple processors are present.

The DTA runs as its own executable outside the SQL Server process space. The executable is called DTAShell.exe. The DTA is focused on the database's physical aspects. If the physical structures are optimized, the query processor will be that much more efficient. The physical performance structures include clustered indexes, nonclustered indexes, indexed views, and partitions.

Current Activity Window (SQL Server Management Studio)

The Current Activity window in SQL Server Management Studio graphically displays information about

- Current user connections and locks
- Process number, status, locks, and commands that active users are running
- Objects that are locked and the kinds of locks that are present

If you are the system administrator for the database, you can view additional information about a selected process or terminate a selected process. The Current Activity window is limited at the database level. If you want to monitor all the databases found on a server at once in an aggregated manner, you have to use a third-party tool or develop one using the SMO.

Event Notifications and Reactive Monitoring

An event notification is a new kind of database object. It executes in response to various DDL statements and trace events. When a notification executes, it sends an XML-formatted message to a Service Broker service. An event notification is similar to a trigger in that it runs in response to an event. Unlike a trigger, however, an event notification is decoupled from the event source; the event message can be consumed asynchronously by receiving messages from the queue for the service.

You can use event notifications to react to database schema changes or any other changes related to database objects. If you take advantage of the Service Broker queuing and delivery support, event notifications can be a powerful ally. Event notifications are worthy of an entire chapter. I recommend that you read about them on Books Online before getting started. It is important, though, to understand the difference between event notifications and triggers. Table 3-1 illustrates the major differences.

Table 3-1 Comparison of Event Notifications and Triggers

Trigger	Event Notification
Responds to DML and DDL events.	Responds to DDL statements and a subset of SQL Server trace events.
Runs SQL or CLR code.	Doesn't run any code—simply sends an XML message to a Service Broker service. However, the service can be designed to activate a stored procedure that processes the message.
The event must be consumed synchronously.	The event is consumed asynchronously.
The event consumer is tightly coupled to the event producer.	The event consumer is decoupled from the event producer.
Events must be consumed on the local server.	Events can be consumed on a remote server.
Executes in the scope of the enclosing transaction.	Does not execute in the scope of the enclosing transaction.
ROLLBACK is supported.	ROLLBACK is not supported.
DML trigger names are schema-scoped.	
DDL trigger names are database- or server-scoped.	Event notification names are scoped by server, database, assembly, or a specific object within a database.
The DML trigger has the same owner as the table for the trigger.	Event notification on an object may have a different owner than the object that the notification monitors.

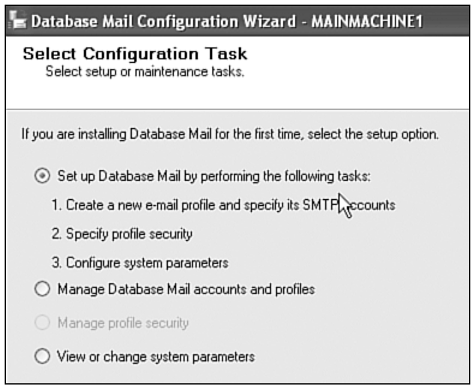
Trigger	Event Notification
Metadata:	Metadata:
select * from sys.triggers	select * from sys.event_notifications
select * from sys.server_triggers	select * from sys.server_event_ notifications
EXECUTE AS is supported.	EXECUTE AS is not supported for the event notification.
DDL trigger event information is available as XML through the new EventData() built-in function.	Sends XML-formatted event information to a Service Broker service. The XML has the same schema as that which is emitted by the EventData() built-in function.

As you can see, the differences between the two choices are significant. In many cases triggers are the best mechanism. The use of event notifications does have an important place. In organizations with server farms, where changes to system structures need to be coupled with operating system changes, the event notification capabilities can be hooked up to a Windows application via WMI. This is pretty advanced functionality. You should research and understand it well before implementing it.

Database Mail

In SQL Server 2005, a new off-by-default technology has been built on the SQL Service Broker. Database Mail replaces SQL Server 2000's SQLMail. Database Mail provides mail-sending capabilities that can be invoked via a user interface or a command-line/code interface. It uses SQL Server Service Broker queuing to send messages.

Database Mail is found in the Management folder of the database explorer for each database instance. It is not included in SQL Server Express, so you can't host a spam server! You configure Database Mail using the Database Mail Configuration Wizard. Figure 3-7 shows this wizard's start page to give you an idea of the parameters you must supply. Notice that Database Mail uses SMTP mail protocols. Before you can use Database Mail, you have to enable SMTP on the server and change the default security settings in Windows. Accomplish this by either continuing through the wizard or going to the Surface Area Configuration (SAC) tool and turning on Database Mail.



Database Mail Configuration Wizard - MAINMACHINE1

Select Configuration Task
Select setup or maintenance tasks.

If you are installing Database Mail for the first time, select the setup option.

[✓ Set up Database Mail by performing the following tasks:]
1. Create a new e-mail profile and specify its SMTP accounts
2. Specify profile security
3. Configure system parameters

Manage Database Mail accounts and profiles

Manage profile security

View or change system parameters

Figure 3-7 Database Mail Configuration Wizard.

Database Mail can be used in a number of scenarios. Here are a couple cases:

- A database administrator has set up a special routine that sends him or her an e-mail when the backup job has successfully or unsuccessfully completed.
- A DBA has set up a routine that sends an e-mail when the Server Processor is at better than 75% utilization.

Database Mail is a send-only program, so it's ready to receive mail and perform an action. Overall, database mail is much safer and easier to manage than mail capabilities provided in early versions of SQL Server.

Managing Very Large Databases

Because scalability is composed of many things, designing for scale is difficult, especially for applications that come packaged from software providers, such as SAP and Siebel. In SQL Server 2005, a number of features provide mechanisms for increasing scalability for Very Large Database (VLDB) systems. The fact is, it's significantly more complex to scale a 500GB database than a 500MB database. The processing overhead for backup and recovery functions can't impede system availability. SQL Server 2005 enables new scalability features that include a horizontal partitioning technology, new methods for backing up and restoring large data sets, and providing higher levels of data reading concurrency. This section covers the most significant and complex VLDB technology: the table partitioning functionality.

Table and Index Partitioning

In previous releases of SQL Server, partitions were created through Distributed Partition Views (DPVs). DPVs were neither easy to set up nor easy to maintain. In SQL Server 2005, DPVs are still available, but they are deprecated. How can you implement partitioning?

Table partitioning is broken into several steps. First, you determine if a table should be partitioned. This is the most important step. Not every table benefits from partitioning. Determine which tables are performing poorly. Take normal corrective action, such as optimizing the indexes. At this point, it would be instrumental to run the DTA against the table to see if it recommends a partition. Also, look at the data. Does the table contain a mix of older data and new data? Find out if there are regulatory reasons for keeping this data live in the system.

After choosing a table, you need to define a partitioning key and decide on the number of partitions. This is trickier; the partitioning key is used to generate the partitioning function. The column used for partitioning should be able to be broken into ranges. Additionally, the range of values ultimately determines how many partitions your table will support. The maximum number of potential partitions is 1,000. Simply stated, the partitioning function maps each row to its appropriate partition. When determining your partition column, consider the total plan. The number of subsets provides the realm of possibilities for long-term partition maintenance. One of the most common partitioning columns is a data column. It provides the most natural means for dividing table data. Once you decide on a partition column, you must design a partitioning scheme.

The partition scheme maps each partition specified by the partition function to a filegroup. Essentially, the partitioning scheme maps the partition to a physical location. Planning the partition scheme essentially involves deciding which filegroup(s) you want to place your partitions on. The primary reason you may want to place your partitions on separate filegroups is to ensure that you can perform backup operations on partitions independently, because you can perform backups on individual filegroups. Additionally, you want to align your data by placing indexes on the same filegroups as the partitioned data. When you align your indexes with your partitioned data, maintenance and query performance are improved. Also, remember that partition schemes are logically separate from the partition function; you can have multiple schemes. Your scheme and function should have the same number of partitions.

Creating a filegroup is the next step, and it requires you to think about hardware. For performance and easier maintenance, filegroups should make it easier to separate the data. The number of filegroups may be limited by hardware resources. Generally, it's best to have filegroups on different spindles so that disk I/O issues are avoided. Separating the data also has a performance benefit, because parallelism is increased across partitions. It's also worth considering whether your partitions allow different quality and quantity of disks. For example, if your system uses a RAID 10 disk array, you might consider keeping the hottest data on those disks. Doing so has many benefits. You might also consider using less-expensive disks for partitions and filegroups of older data that doesn't have significant workload pressure. With the portioning scheme and column planning completed and the filegroups decided on and created, you can focus on the task of creating the partitioning function and scheme and partitioning the data. When you create the partitions, remember that you must consider two boundaries: the left and the right. The partitioning function must include all data and should be restricted through a check constraint.

When you actually create and set up partitioning, you must create new tables. This may be a problem in systems where partitioning is needed but you can't rebuild the table structure. Partitioning has some other limitations, such as data type limits. You cannot use SQL CLR, timestamp, image, or ntext types as the partitioning column. The columns must be deterministic and persisted in the column, so you can't use a derived column. This affects mostly data warehouses, so for non-data-warehouse usages, these barriers should not be a problem. Finally, the partitions must be on the same node.

In addition to partitioning table data, indexes can be partitioned. Secondary indexes can be set up completely separately from primary indexes. The syntax for creation is the same. When the indexes and partitions are within the same filegroup, the indexes are aligned. Alignment provides several advantages; most importantly, it provides a means for simplifying data backup. Query performance is better in aligned index systems, because the I/O aspects of query processing are increased.

Backup and Restore Enhancements

For database administrators, the most gut-wrenching experience is being called in to back up from a media set for a database and having the media fail. Moreover, in previous versions of SQL Server, you couldn't mirror the backups, so you had only one set of backups to work with. If there was a disaster and the backup was lost, the data was gone forever. SQL Server 2005 has new check features for ensuring the quality of the backed-up data.

Checksum Integrity Checks

SQL Server 2005 introduces a dbcc_checksum statement that enables extra data verification. The checksum is enabled using the SET page_verify recovery option of the alter database command. The page_verify command provides three options to discover incomplete I/O transactions caused by disk I/O errors:

- **Torn page detection.** If this option is specified, a bit is reversed for each 512-byte sector in the 8KB database page when the page is written to disk. If a bit is in the wrong state when the page is later read, the page was written incorrectly, and a torn page is detected. This is the default option.
- **Checksum.** If this option is specified, a checksum is taken over the contents of the entire page and is stored in the page header when a page is written to disk. When a page is read from disk, the checksum is recomputed and compared to the checksum value stored in the page header. If the values don't match, an error message is reported to both the SQL Server error log and the NT Event Viewer.
- **None.** If this option is specified, the page_verify_option is set to OFF. Future data page writes will not contain a checksum, and checksums will not be verified at read time even if a checksum is present.

Disk I/O errors can cause database corruption that is often the result of a power failure or a disk hardware error that occurs when data is being written to disk. The CHECKSUM option provides the most comprehensive level of integrity checking, offering an extra level of protection for detecting disk I/O errors that may not be detected by the disk hardware itself.

Fast Recovery

SQL Server 2005 improves the availability of SQL Server databases with a new, faster recovery option. Users can reconnect to a recovering database after the transaction log has been rolled forward. Earlier versions of SQL Server required users to wait until incomplete transactions had rolled back, even if they did not need to access affected parts of the database. A new database option, ALLOW ACTIVITY DURING UNDO, is turned on by default.

Online Restore

SQL Server 2005 introduces the ability to perform a restore operation while an instance of SQL Server is running. Online restore improves SQL Server's availability, because only the data being restored is unavailable. The rest of the database remains online and available. Earlier versions of SQL Server required that you take a database offline before performing a restoration.

You can choose between two options when using online restore:

- An online file-level restoration of an entire database file
- An online page restoration of a single page of data

SQL Server 2005 also supports the online restoration of a filegroup, because a filegroup is nothing more than a collection of files.

Mirrored Backups

SQL Server 2005 introduces support for mirrored backup sets, which increases the reliability of SQL Server backups. Earlier versions of SQL Server supported only a single copy of a given backup. If backup media were damaged, roll-forward would take longer or fail. In SQL Server 2005, backup media can now be mirrored. For example, an administrator can set up four tape devices to back up two media families, with a mirror for each media family. The corresponding volumes in each mirror have identical content, making them interchangeable at restoration time. Administrators can implement up to four mirrored backup sets.

Full-Text Catalog Inclusion

SQL Server 2005 provides integrated backup and restore facilities for full-text catalogs. Earlier versions of SQL Server did not provide an integrated and reliable mechanism by which full-text catalogs could be backed up and restored. In SQL Server 2005, full-text catalogs can be backed up and restored along with, or separate from, database data. This functionality reduces the time needed to recover from a disaster and simplifies the task of moving data, including catalogs, from one computer to another without the need to fully repopulate the catalog.

This backup and restore feature provides the following capabilities:

- You can back up and restore one or more full-text catalogs to and from media in the same manner as other data.
- It eliminates the need to fully repopulate data after a restoration.
- It updates full-text data to reflect changes by rolling logs forward after a restoration. Change tracking must be enabled for this capability to work.

SQL Server Replication Enhancements

SQL Server Replication enables synchronization of data in distributed and mobile deployments, supporting application in all lines of business, from e-commerce to customer relationship management. Microsoft has shipped replication with previous versions of SQL Server.

NOTE This section doesn't cover all the uses and features of replication; it concentrates on the new features. SQL Server Books Online has deep and exceptional coverage of replication.

SQL Server 2005 Replication Services includes the following new features and improvements:

- **Programmability.** A set of CLR classes is provided for configuring, managing, scripting, and monitoring replication. You can use these object models to programmatically control replication objects such as publication and subscription.

- **Manageability.** Many enhancements to the setup and management of replication topologies are introduced. These include multiple subscription creation with the Create Subscription Wizard, a lightweight merge Subscriber option that stores much less metadata, and an enhanced ability to do a transactional subscribe from a backup. Also, the Replication Monitor is enhanced to help you oversee replication operations across your entire enterprise from one console. The Replication Monitor has also been redesigned so that users can better understand the state of replication actions.
- **Availability.** Improvements to snapshot generation, schema replication, and metadata maintenance are introduced. Snapshots interrupted during delivery can automatically resume without resending files that already completely transferred. Also, a much broader range of Data Definition Language (DDL) changes can now be replicated without using special stored procedures. Finally, Merge Replication now features automatic retention-based metadata cleanup, providing easier maintenance of merge topologies.
- **Business intelligence/data warehousing.** Support for Oracle as a first-class transaction replication publisher is introduced. Oracle data can be published directly to SQL Server, where you can take advantage of the business intelligence and data warehousing tools and technology provided by SQL Server and the .NET platform.
- **Mobility.** Message-based replication is introduced, which is the ability to replicate data over the HTTP or SOAP protocol. Message-based replication is well suited to replication over the Internet and for topologies that include mobile subscribers such as Microsoft Windows CE devices.
- **Scalability and performance.** New performance optimizations are introduced, including partition groups for filtered merge replication publications and new article types for customers who want little or no change tracking on certain portions of their publication. Also, transaction replication performance is improved by using multiple connections to retrieve and apply commands.

Scenario: Replication Management in a Complex Topology

Suppose you work for a large company and are responsible for managing replication of SQL Server databases. You set up transaction replication between servers at the corporate office so that the log on the Publisher is read and transactions are distributed to subscribers continuously. You also set up merge replication to publish product and inventory information from corporate servers to the sales force and to replicate orders from the sales force back to the corporate servers. After the publications and subscriptions have been created, you use Replication Monitor to define the maximum time to allow data to be replicated between servers. You leave Replication Monitor running in the background so that you can check it from time to time to be confident that data is flowing smoothly and quickly. When a problem occurs, Replication Monitor alerts you, and you drill down to find the specific problem and fix it.

The SQL Server 2005 Replication Monitor is enhanced to help you oversee replication operations across your entire enterprise from one console. Replication Monitor helps you discover performance problems before they become critical. When problems occur, the new Replication Monitor provides more detailed statistics and history to help you troubleshoot and solve problems faster.

Replication Monitor allows you to do the following:

- View a summary of replication activity across an enterprise in one display and drill down to more detailed information as needed.
- Set warning levels so that Replication Monitor can alert you when the delivery of data to subscriptions takes too long.
- Leave Replication Monitor running in a small background window that pops up notifications when synchronization slows down or an error occurs.
- Control synchronization schedules, properties, and notifications from a central location.
- Receive notification through a replication alert when an event occurs on a replication agent.
- Validate subscriptions to ensure that data values are the same at the publisher and subscribers.
- Reinitialize one or all subscriptions to a publication as needed.

Peer-to-Peer Replication

Peer-to-peer transaction replication is designed for applications that might read or modify the data at any of the databases participating in replication. For example, an online shopping application is well suited for peer-to-peer replication. You can improve application performance by spreading out queries that read data across multiple databases. Additionally, if any of the servers hosting the databases are unavailable, the application can be programmed to route traffic to the remaining servers, which contain identical copies of the data. Read performance is improved because activity can be spread across all nodes. Aggregate update, insert, and delete performance for the topology is similar to a single node, because ultimately all changes are propagated to all nodes.

All nodes in a peer-to-peer topology are peers: each node publishes and subscribes to the same schema and data. Changes (inserts, updates, and deletes) can be made at all nodes. Replication recognizes when a change has been applied to a given node, preventing changes from cycling through the nodes more than once.

Two basic topologies are recommended for use with peer-to-peer replication:

- The first topology is typical of an e-commerce or web-based application platform. In this case, each peer node server has a database that participates in the transaction replication modality. Microsoft recommends dividing the workload among the peers so that one is the read server and the other is the write server. The writes are passed to the read server. Although both servers can function as a read database, transaction replication requires that only one database receive writes.
- The second topology is one in which the peers span a significant time zone difference, and thus no overlaps occur in write activity. In this case, all the peers receive the transaction updates across the topology and, due to the difference in time zone and business hours, no contention occurs. Each database is independent from the other. Because the databases' peak load hours are isolated, transaction contention is minimized.

Oracle Publication

With Microsoft SQL Server 2005, you can include Oracle Publishers in your replication topology, starting with Oracle version 8.0.5. Publishing servers can be deployed on any Oracle-supported hardware and operating system. The feature is built on the well-established foundation of SQL Server snapshot replication and transaction replication, providing similar performance and usability. The main purpose of Oracle as a publisher is to get data out of Oracle and into SQL Server Analysis Services. In many customer scenarios, legacy data used for reporting resides in Oracle. The Oracle publishing functionality is focused on a few scenarios and is not an attempt to duplicate Oracle functionality. If you're considering using the Oracle Publisher feature, it's highly recommended that you read the Books Online topics.

Web-Based Replication Update

One of the features that many replication customers asked for was the ability to perform merge replication via Hypertext Transfer Protocol/Secure (HTTP/S). The ability to have a subscriber request a merge through port 80 allows for customer scenarios wherein a VPN connection is not possible. The Internet Server Application Programming Interface (ISAPI) service on the server acts as a storage location for articles that are exchanged in merge replication. Simply put, replication provides the data to a virtual server in Internet Information Services (IIS), which the client application (typically a Windows Pocket PC or laptop computer) then connects to. The connection is accomplished over the same protocols as any other HTTP command. The client application use scenario is one in which the user of the database is not connected to the main data store. The user typically modifies the data on his or her local source and updates the main database upon connection/synchronization. For example, a salesperson may keep her list of customers on a local machine and add new customers and orders while she is in the field. At the end of a period, she synchs the data by connecting her laptop to the corporate back-end data.

Performance and Scalability

Both merge and transaction replication provide new features for increased performance. Replication has three main areas of focus to improve performance and scalability. Both merge and transaction replication have increased support for more concurrent subscribers (greater concurrency is one area of improvement). With more subscribers, the ability to scale through higher levels of throughput is expected through new features—the second area of improvement. The third area of improvement is better control and design through the ability to define package sets in a more granular fashion. Table 3-2 describes the new performance and scalability features.

High Availability for the Masses

Changes in the IT world have dictated new definitions of availability. Once upon a time there was an acceptable amount of planned downtime for system maintenance. With businesses being open 24/7, however, system maintenance windows are being squeezed. Moreover, unplanned downtime is catastrophic for organizations whose livelihoods depend on the availability of the IT infrastructure. This is nothing new. SQL Server 2005 delivers new technologies for availability and enhancements to already-common technology.

High-Availability Solutions

SQL Server gives you many approaches to creating high availability. There really isn't a silver bullet. Looking at the landscape of customers, you can find that each application and scenario has required much creativity on the part of the IT teams. Microsoft has improved its core high-availability (HA) technology, called failover clustering. Microsoft introduces a new technique for log shipping: database mirroring. Log backup shipping is still supported, but little significant work has been done. Table 3-3 examines the three core technologies and shows how they affect overall availability. This table is not exhaustive, but it covers the meat of the issue. Remember that one of the goals of HA solutions is to mask hardware and software failure. Database mirroring and failover clustering are completely different in that way. A failover clustering solution requires two identical systems. Database mirroring runs on commodity hardware. Log backup shipping is a warm standby solution that really only provides a solution when geographic distance provides an added level of security.

Table 3-2 New Performance and Scalability Features

Merge Replication
Concurrent phase agents: Merge replication performance is increased by running the download phase and the upload phase of the merge agent in parallel. This is particularly beneficial when the merge agent is being run on a local area network (LAN) or is using the high-volume server-to-server profile. New download-only articles: Article types provide a new performance optimization for customers who only have data that is changed only at the publisher. New well-partitioned articles provide a new performance optimization for customers who make changes to only their subscribing partition. Because the data is well partitioned, internal steps of the merge reconciliation process can be skipped, providing a performance improvement during the upload phase. Partition groups are a new performance optimization for filtered merge replication publications. When a subscriber synchronizes with a publisher, the publisher must evaluate the subscriber's filters to determine which rows belong to that subscriber's data set. Without merge partition groups, partition evaluation must be performed for each change uploaded to the publisher since the last time the Merge Agent ran for a specific subscriber. If the publisher is running on SQL Server 2005 and uses merge partition groups, each subscriber evaluates only the changes that meet the subscriber's filter criteria. This can lead to significant performance gains when a publication has a large number of changes, subscribers, or articles in the publication. There are some new restrictions on filters when using merge partition groups. But if these restrictions are met, performance can be significantly improved.
Transaction Replication
Multiple distribution streams performance is improved by using multiple connections to retrieve and apply commands. This is particularly beneficial when the Distribution Agent has not run for a long period of time or if a large spike occurs in the number of commands to be replicated.
Snapshot Replication
Parallel snapshot preparation processes several articles while scripting schema or bulk-copying data within the Snapshot Agent. Administrators do not need to configure any special options to take advantage of this new functionality, because it is now embedded in the process of preparing snapshots. This feature allows SQL Server to prepare snapshots with greater speed and efficiency than was possible with earlier versions of SQL Server.

NOTE Although Microsoft might disagree, I don't include replication in the matrix of HA technologies because replication doesn't provide an automatic means for failure detection. Additionally, it doesn't supply any means for recovering or reversing errors. What happened with replication is that many companies use it to distribute copies of the data to web farms that have many read-only catalogs of data. This web farm of servers in a sense masks a single point of failure. After all, if server A fails, and it's sharing work with servers B and C, the remaining two servers should pick up the workload. That doesn't make it a real HA solution; it makes it a load-balancing solution.

Table 3-3 Comparison of High-Availability Solutions

Availability Feature	Database Mirroring	Failover Clustering	Log Backup Shipping
Standby type	Instant	Hot	Warm
Failure detection	Yes	Yes	No
Automatic failover	Yes	Yes	No; Network Load Balancing (NLB) helps
Masks disk failure	Yes	No; shared disk	Yes
Masks SQL process failure	Yes	Yes	Yes
Masks other process failure	No	Yes	No
Metadata support	Database	All system and database	Database
Transactional consistency	Yes	Yes	Yes
Transactions are current	Yes, always up to date	Yes, always up to date	No, since last log backup
Perceived downtime	Seconds	30 seconds plus database recovery time	Seconds plus database recovery time
Transparent to client	Yes; auto-redirect	Yes; reconnect to same IP	No; app must know standby

Availability Feature	Database Mirroring	Failover Clustering	Log Backup Shipping
Special hardware needed	No; duplicate system needed	Specialized hardware from cluster Hardware Compatibility List (HCL)	No; duplicate system needed
Distance limit	Virtually unlimited	100 miles	Dispersed
Complexity	Some	More	Some
Standby-accessible	Standby-accessible; some performance impact	Standby never accessible	Yes; multiple copies, read-only, percentage depends on update frequency
Impact on performance	No impact to minimal impact	No impact	Minimal; file copy on primary
Impact on backup strategy	No impact	Must be able to back up from any node	Minimal; many small backups

Additionally, you should consider the operating system when planning your HA solution. Table 3-4 shows the differences between Standard Edition and Enterprise Edition.

Table 3-4 Differences Between Standard Edition and Enterprise Edition for HA Solutions

HA Technology	Standard Edition	Enterprise Edition
Failover clustering	None	Up to eight nodes
Database mirroring	None	Available
Database Snapshot	None	Available
Multi-instance	16 instances	50 instances

Failover Clustering

SQL Server 2005 failover clustering provides HA support for server-wide failure. With failover clustering, the operating system and SQL Server work together to provide failure protection by providing redundant hardware and an automated mechanism to move the database server to secondary hardware if the primary fails. Failover clustering supports up to eight nodes, depending on which edition of Microsoft Windows Server 2003 is running on the server.

SQL Server has extended the capabilities of failover clustering to SQL Server Analysis Services, Notification Services, and SQL Server Replication. With SQL Server 2000, SQL Server Agent and other job management and processing capabilities were not covered by failover clustering. Now, the technologies are clustering-aware. SQL Server failover clustering is now a more complete server-level redundancy solution, although clustering solutions are difficult to implement well and are expensive. For most customers, failover clustering is used only with their most valuable transactional databases.

Looking back at Table 3-3, you see a couple of failover clustering features that make it the best HA technology. First, failover clustering participates at the operating system level. This means that failover clustering is aware of operating system process failures. Some operating system-level processes may interfere with SQL Server or block it from answering a connection request. Moreover, failover clustering covers all the databases on the server that are participating in clustering. This means that the master is protected. Additionally, SQL Server 2005 has a hidden database called System Resource that is covered in clustering but that is inaccessible in database mirroring.

On the other hand, having two synchronized servers, with one doing "nothing," is very expensive and difficult to manage. As I've said before, if it's mission-critical, it has to be failover-clustered. Other solutions will just prove to be a nightmare. Another thing worth noting is that all the "goods" (so to speak) are in Enterprise Edition, so the best use of that level of licensing is failover clustering.

Database Mirroring

SQL Server introduces a new technology set that allows you to create a hot standby database that maintains close synchronization with a primary database. Database mirroring lets you create hot standby databases that provide rapid failover with no loss of data saved by committed transactions. You can think of database mirroring as real-time log shipping. If the primary system fails, applications can reconnect to the database on the secondary server almost immediately, without waiting for recovery to finish.

Database mirroring capabilities are affected by which edition of SQL Server you are running. If you choose to use mirroring with Standard Edition, your failover options are limited, as shown in Table 3-5. That said, if you can live with manual failover, Standard Edition may suffice. Although Developer Edition has all the features of Enterprise Edition, it is not licensed for use in production environments. Any of the SQL Server editions can be a witness, including Express Edition.

Table 3-5 Data Mirroring Capabilities

Feature	Standard Edition	Enterprise Edition
Partner	Yes	Yes
Witness	Yes	Yes
Full safety	Yes	Yes
Safety off (High-Performance mode)	No	Yes
Parallel redo (using multiple threads to replay transaction logs)	No	Yes
Database Snapshot	No	Yes

Of these differences, the most notable is the lack of Database Snapshot, which doesn't affect the deployment of mirroring. Also, parallel redo may affect large databases. Interestingly, Standard Edition doesn't support high-performance mode. Microsoft may have left many of the mirroring features out of the Standard Edition simply to sell more Enterprise Editions because high-performance mode is preferable in a number of scenarios. I'd include any application that has high transaction volumes and applications wherein geographic distance may allow the business to have acceptable levels of loss/redo.

Here are some cases in which you might need to use database mirroring:

- An administrator needs to improve the availability of a SQL Server database that supports several critical applications. The administrator wants solutions that automatically fail over quickly, require no shared storage components or special controllers, and automatically resynch with a primary database after a failover. By configuring database mirroring with primary, secondary, and witness servers, the administrator can implement a system that automatically fails over in the event of a server failure or lost connection to the primary database.
- An administrator wants to protect non-mission-critical but important databases. Database mirroring provides a good solution for Notification Services, Reporting Services, and SQL Server Service Broker databases.
- If you design your application to use many smaller Service-Oriented Architecture (SOA) databases, using database mirroring provides a cost-effective means to daisy-chain the servers, providing a revolving series of failover database servers.

At this point it's really important to understand the application and database implications of database mirroring. Database mirroring is a single-database technology. Failover clustering by design includes all the system databases, users, roles, and SQL Agent jobs, just to name a few, but database mirroring does not.

If your application uses linked servers to make cross-database calls to objects outside the mirror database, the application is not a good candidate for mirroring. When you are looking at database mirroring, spend a significant time understanding the database scalability scenarios. Although it's true that mirroring doesn't require exactly the same hardware, not using parity servers can cause possible issues with customer satisfaction and performance.

Database Mirroring Concepts and Confusion You must master a number of new vocabulary words and concepts. At the most basic level, database mirroring requires only two "partners"—the principal and the mirror. But, depending on your needs, you may also choose to have a witness server. Moreover, you need to make a decision about safety of transactions versus performance. Let's look at the basic concepts in depth.

A database mirroring system requires three servers that are running SQL Server. Each server has a specific architectural role:

- **Principal.** The server on which applications connect and where transactions are processed.
- **Mirror.** The target of transaction log records, which can be applied either synchronously or asynchronously. The mirror server exists in a state that does not allow direct read access to the data. As transaction log records are generated on the principal server, they are continuously replayed on the mirror server, which produces a state in which the mirror server is normally behind the principal server by only the time it takes to replay the log written in a single log write. This provides a duplicate of the data at a point in time.
- **Witness.** An arbiter within the architecture, providing the tie-breaking "vote" in determining which server is the principal and which is the mirror. Two servers in the architecture must agree in order for a server to be designated the principal and thus the target of all transactions. From a client application point of view, the failover from one server to the next is automatic and nearly instantaneous. The witness server is needed only when automatic failover is needed. For failover to happen automatically, the partners (the mirror or principal and the witness) must reach a quorum. When the principal fails to respond to the ping from the witness, the mirror is promoted to principal and takes over the job. However, the witness doesn't e-mail the DBA that the principal is down!

One of the most important things about database mirroring, besides the extremely low latency provided during a failover from the principal to the mirror, is the capability to synchronize changes in both directions. If the principal goes offline and the application fails over to the mirror, the mirror becomes the principal server in the architecture. When the failed server comes back online, it is designated the mirror. Transaction log records from the principal are applied to it to bring it into synchronization with the state of the database at that point in time.

Databases participating in database mirroring have three modes of partnering. As an administrator, you can choose between High-Availability mode, High-Protection mode, and High-Performance mode. In High-Performance mode, transactional safety is turned off. Basically, this means that when set to “off,” the principal does not wait for the mirror to acknowledge receipt or hardening of a transaction. In this case, the principal throws the transaction over the wire and continues. Depending on volume, this could mean a little latency or a lot. During testing, you should try to understand the risk level.

In High-Protection mode, transactional safety is ensured by having the principal wait for the mirror to acknowledge receipt of transactions. This is similar to High-Availability mode, with one exception. High-Availability mode provides the possibility of automatic failover, whereas High Protection does not. You can look at database mirroring as a synchronization technology too. By that, I mean for High-Availability and High-Protection modes, transactions are synchronized across the servers; in High-Performance mode, they aren’t necessarily synchronized.

When getting ready to use database mirroring, you have some important considerations:

- The principal database must use the FULL recovery backup model. This means that the principal should be doing a lot of bulk insert work, because the log files resulting from bulk operations aren’t shared with the mirror. Why? Too much bandwidth and log file traffic would keep mirroring from working well.
- The mirror database must be initialized from a current backup of the principal database using the NORECOVERY mode, followed by restoring the transaction logs sequentially from the principal’s transaction log. This process ensures data consistency and synchronization of the initial term of the relationship. Without this, the mirror catches up too much.
- The database names for the mirroring partners must be the same.
- You cannot directly access the mirror during the partnership session. Use a Database Snapshot if you need to access the database.

Database Mirroring States at a Glance Database states are kept during the database mirroring session. A session begins when two servers are partnered. The state can be found in the sys.database mirroring catalog view. You can review all the state numbers and their descriptions by reading about the topic in Books Online. Moreover, several server concepts are noteworthy. These phrases help you understand how the session is doing:

- **Exposed** is when the principal is processing transactions but no log data is being sent to the mirror. When this state occurs, mirroring doesn't work. Essentially, you are not in a protected state from a transactional point of view.
- **Cannot serve the database** is when the principal doesn't allow any user to connect to the database or any transactions to be processed.
- **Isolated** is when a server cannot contact any of the other servers in the mirroring session, and they cannot contact it. This state is really a lack of communication.
- **Synchronized** is when safety is set to full, the records are successfully passed to the mirror, and the mirror hardens the logs.
- **Synchronizing** occurs when safety is set to OFF and the logs are always in a catch-up state.
- **Suspended** is the state that occurs when the session is broken. When the session is paused or the mirror has redo errors, the session state is set to SUSPENDED.

For a complete and thorough discussion of database mirroring and states, see Books Online.

Automatic Client-Side Redirect One of the key ingredients of database mirroring success is the use of automatic client redirect. This is a new technique for caching the location of the principal and mirroring for client connections. You can use client redirect with only the new data access protocols: ADO.NET 2.0 and SQL Native Client. It's relatively simple to use: In the connection string, you provide the name or IP address of the principal and mirror databases. When a client connects, the client access code checks to determine which machine is available and then connects appropriately. (Here's an example of a connection string: "server=Partner_A; failover partner=Partner_B; database=AdventureWorks".) For more information about automatic client redirect, see SQL Server Books online and search for the topic "Client Connections to a Mirrored Database."

So that using database mirroring is extremely easy, set up a mirroring system via the Database Properties dialog box. (See Chapter 2, "What Everyone Should Know About Security," for security issues related to database mirroring.)

NOTE Automatic Client Redirect is available in Enterprise Edition only.

Log Backup Shipping

Log backup shipping has an interesting history. It was possible to do log shipping with SQL Server 2000. Originally Microsoft didn't support it, but customer demands for a warm standby technology pushed the technology into a supported mode.

In reality, log shipping is really a backup log-shipping technology. Log shipping allows you to automatically send transaction log backups from one database (known as the primary database) to a secondary database on another server (known as the secondary server). At the secondary server, these transaction log backups are restored to the secondary database, keeping it closely synchronized with the primary database. An optional third server, known as the monitor server, records the history and status of backup-and-restore operations and optionally raises alerts if these operations fail to occur as scheduled.

In comparison to database mirroring, which is a real-time log-shipping technology, backup log shipping is an adequate solution for non-mission-critical applications that can stand some data loss. Additionally, log-shipping servers can be located a great distance apart, and they provide a solution in situations where Mother Nature may cause a headache. With SQL Server 2005, little work was done to increase this feature's performance or reliability. In SQL Server 2005, you cannot upgrade log shipping from SQL Server 2000. Instead, you have to migrate the log-shipping databases by applying the upgrade to each server—the primary and the secondary—while each is offline. (You can read about how to accomplish this by visiting SQL Server 2005 Books Online and searching for the topic "Upgrading a SQL Server 2000 Log Shipping Configuration.") You can also set up log shipping by opening the Database Properties dialog and, in the Transactional Log Shipping dialog, provide the needed information.

General Data Availability

Microsoft has two new features that help applications connect to the database, which is referred to as general database availability:

- **Table partitioning.** Table partitioning is the ability to break a table into smaller subtables and then use them as if they were one table. This allows for better query performance and a more manageable database.
- **Snapshot isolation.** This is a new transactional locking scheme that keeps readers from blocking writers and gives data users a consistent point-in-time view of the database. This implementation of snapshot isolation overlaps with the ANSI standard for isolation levels.

Before we delve into the details of table partitioning, you should have a better understanding why partitioning is interesting. Going back to basic database development, database tables are designed to store all an entity's attributes. For example, your customers table contains the relevant customer information. In the sales table, the order header and order detail tables together contain the relevant characteristics of an order. Over time, these tables become large. Also, the tables' access patterns are inconsistent. The customer table receives relatively fewer updates than the order detail table. Additionally, organizations are now finding that the use of business intelligence and reporting are requiring older data to be kept in active systems for longer periods of time. After a point, the sheer size of the table is detrimental to query performance. So what makes a table large? The number of rows and the number of bits per row. The first factor is consequential to the query processor. You might conclude that a large table is one that doesn't perform well. You might also add that a table that requires extended periods of time to properly maintain the indexes could be a large table. The simple goal of partitioning is to make the table more manageable by breaking it into smaller pieces. SQL Server 2005 supports only one kind of partitioning—range partitioning. A type of partitioning called vertical partitioning breaks columns into partitions; this is supported through distributed partition views. Testing and using partitioning has many benefits:

- Greater ease of management. Sliding window partitions can be added and removed.
- Better query performance for partitioned tables via increased parallelism on multi-CPU systems.
- Large-scale deletions and insertions can be greatly improved.
- Creation of relational data warehouses is simplified.

Snapshot Isolation

Snapshot isolation provides a locking mechanism that prevents readers from blocking writers on systems with a high level of mixed workload. In a perfect world, data would be divided by usage: a read database and a write database. This seems logical as a model, but in fact it is difficult to do. Transactions systems focus on inserts and updates, using stored procedures or dynamic Transact-SQL to insert new information. The read-only-type database is heavy on indexes and views, with queries that require significant caching and computational horsepower. Snapshot isolation is implemented as an optimistic locking strategy. DBAs have fine-grained control, allowing for usage of snapshot isolation at the query or stored procedure level. Snapshot isolation needs to be turned on at the database level. Inside SQL Server, snapshot isolation creates a read-consistent version of the data and places it in the TempDB.

Here are some cases in which you might need to use snapshot isolation:

- A large retail company records all transactions in a database and updates inventory levels in real time. Employees need to execute reports and ad hoc queries against this database. The DBA can set the isolation level to Snapshot to allow the reporting and queries to run in parallel with the inventory application.
- A DBA needs to increase the consistency of aggregates such as AVG, COUNT, and SUM, index intersections, and index joins without escalating read scans to a higher isolation level.

Snapshot isolation uses a row-versioning mechanism that stores the version of the data being queried at the time of the connection. The SQL-99 standard defines the following isolation levels, all of which SQL Server supports:

- Read uncommitted (the lowest level, where transactions are isolated only enough to ensure that physically corrupt data is not read)
- Read committed (Database Engine default level)
- Repeatable read
- Serializable (the highest level, where transactions are isolated from one another)

The new implementation of read-committed isolation level uses row versioning if the READ_COMMITTED_SNAPSHOT database option is set to ON. If the option is set to OFF, the read-committed isolation level uses the same type of locking as in earlier versions of SQL Server.

Indexed View

An Oracle feature that Microsoft delivers in SQL Server 2005 is the indexed view. Application concurrency is increased with indexed views. Indexed views are called materialized views in Oracle circles. When a view contains an index, the index is created and stored in a unique clustered index in the same way that a table with a clustered index is stored. Nonclustered secondary indexes can be created after the first unique index is created. Within this new ability, developers can use index outer joins, scalar aggregates, and batches using ROLLUP and CUBE functions. The benefits of indexed view are as follows:

- The overhead of dynamically building the result set for each query that references the view can be eliminated.
- The optimizer can use the view index in queries that do not directly name the view in the FROM clause.
- Existing queries that reference columns in the indexed view can benefit from the improved efficiency of retrieving data from the indexed view without having to be recoded.

Here are some cases in which you might need to use indexed view:

- A frequently executed query aggregates data from several tables and uses an outer join to combine the results. A developer designs a view to provide the same results as the query. With support for indexed views containing outer joins, the developer can index the view and provide a more efficient solution to retrieve the aggregated data.
- You have a database that contains a number of computed columns, or you want to store computed columns in a view. Index view provides better query performance for these objects by use of the index.

Common Language Runtime and Database Administrator

During the beta of SQL Server 2005, I showed the CLR functionality to hundreds of database administrators. Their reactions ranged from "Whatever" to "Don't put that in my database!" This section hopefully will demystify the CLR integration into SQL Server 2005. I would be remiss if I just said "Don't worry." It might be worth demystifying the integration. None of the core functionality of SQL Server 2005 is run as managed code. That said, SMO interfaces are all managed code, SQL Server Management Studio is managed code, and the new Report Builder found in Reporting Services is also managed code. Chapter 4, "Features for Database Development," covers how and when to use the CLR. This section covers the basics of managing assemblies residing in SQL Server.

How Deep Is the Integration?

The Common Language Runtime (CLR) is deeply integrated into the database engine. The CLR is really an engine of sorts that hosts applications—a special kind of application created using managed code. SQL Server is really a cohost of managed code. The SQL Engine works directly with the CLR to manage assemblies that have been placed inside SQL Server and that are called by Transact-SQL or MDX queries. Figure 3-8 illustrates the coupling between the SQL Server Engine and the CLR.

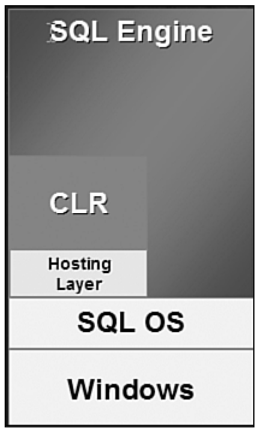


Figure 3-8 The coupling between the SQL Engine and the CLR.

When you create an assembly in SQL Server, the bits are loaded into a table in the database. The assembly is registered but is not automatically pulled into memory. The bits simply exist in the database. The interaction between the CLR/hosting layer and the SQL operating system (OS) occurs until the assembly is called by a procedure. Upon evocation, the SQL Server Engine works with the CLR to manage memory, execution, and destruction of the running code for that assembly. Thus, the CLR goes through the SQL OS for the following:

- **Memory.** SQL Server manages its own memory. The CLR asks SQL for memory as needed.
- **Threads/fibers** get things done. Since SQL Server uses these natively, it makes sense for SQL to manage the CLR. All the threads are managed by SQL Server.
- **Synchronization** is moving data into and out of assemblies to TDS and in memory.

The hosting layer has a set of APIs that manage the communication between the SQL Server OS and CLR. When objects are called, the CLR asks SQL Server for memory buffers, thread allocations, and security. The hosting layer manages the multiple security layers. The CLR uses something called an app domain to create an execution context that is really just a container for all the managed code found in a particular namespace. The app domain provides a container that controls interassembly interaction. Basically, no interaction occurs between different app domains. Thus, if a problem occurs, such as a memory issue, the app domain is the main container for SQL Server to unload the assembly. The CLR manages the escalation policy for assemblies. The CLR also maintains the state of the assembly.

This topic could take up an entire chapter, but briefly, here's what you need to know under All memory allocation from CLR through SQL Server. Most of the CLR memory (GC Heap) comes from a multipage memory allocator (outside Buffer-pool or MemToLeave). For memory-intensive operations, the SQL OS tells the CLR to initiate garbage collection to clear memory. Note that the MAX Memory setting in SQL Server does not cover CLR. This is because the CLR exists in the operating system memory, not the SQL OS. To understand what's happening with the CLR, look at the following DMVs:

```
Sys.dm_os_memory_clerks
Sys.dm_os_memory_objects
```

Most of the interaction between SQL OS and the CLR is thread oriented. The threads are managed by the SQL Server Scheduler, so they are visible to SQL Profiler and Activity Monitor. The SQL Profiler Per counters that are useful for assembly monitoring include # GC and the amount of memory allocated. Look at the memory clerks for SQLCLR and SQLCLRASSEMBLY. The threads are scheduled cooperatively. SQL Server scheduler was designed to punish nonyielding tasks—either managed or Transact-SQL—such that they are forced back. This should prevent a nonyielding task from taking down the server. A task that doesn't yield is called back, stopped, and put back into the queue or forced to miss turns.

Here's a brief list of interesting ways to monitor assemblies:

- Profiler trace events:
  - CLR:load assembly monitors assembly load requests (successes and failures)
  - SQL:BatchStarting, BatchCompleted
  - SP:Starting, Completed, StmtStarting, StmtCompleted monitor execution of Transact-SQL and CLR routines
- Performance counters:
  - SQL Server: Total CLR time
  - .NET CLR Memory
  - Processor

- DMVs and catalog views:
  - sys.assembly* shows basic information about the assemblies stored
  - sys.dm_os_memory_clerks
  - sys.dm_clr*
  - sys.dm_exec_query_stats
  - sys.dm_exec_requests
  - sys.dm_exec_cached_plans

For more insight into when and how to use managed code, refer to Chapter 4 and Books Online. This topic is extremely new to SQL Server, and significant concern was voiced during the beta process, but you have a number of good resources for getting up to speed.

Business Intelligence and the Database Administrator

Chapter 5, "Overview of Business Intelligence," covers the new features of the SQL Server business intelligence (BI) platform. For the average DBA, BI seems like fancy reporting, and actually, it is. But here's the twist: Many of the new functions in Analysis Services, SQL Server Integration Services, and Reporting Services will creep into the DBA's workload. First, I recommend that DBAs read Chapter 5. Many DBAs will find the new Reporting Services functionality exciting and useful, especially those who generate reports for executives. I want to pose some questions for you and your coworkers to think about:

- What impact does Reporting Services (especially the new ad hoc reporting tool, Report Builder) have on your server's security and performance? How much RAM should the server have?
- Should we use 32-bit or 64-bit platforms?
- How many applications do we run in one instance of SQL Server?
- How should we include and manage Analysis Services, Integration Services, and Reporting Services in our disaster recovery plan? With inclusion of these features in high availability, how do we manage complexity?
- With the ability to have an extraction, Transformation, and Loading Server via SQL Server Integration Services (SSIS), how do you build an infrastructure that uses best-of-breed technologies across the company?

Organizations will have to grapple with these types of questions (and many more) as they move to SQL Server 2005.

Summary

For the DBA, little is unchanged between versions of SQL Server. DBAs will welcome many of the time-saving features as well as the new scalability and availability features. SQL Server has improved its portfolio of very large database technologies to increase the availability of large databases. These new technologies create new challenges and opportunities for IT personnel. The role of the DBA is evolving.

This chapter covered the new management tools and .NET assemblies for programmatically working with SQL Server. It also covered new techniques for monitoring and managing SQL Server and looked at features for very large databases, backup and recovery, and SQL Server replication. Finally, this chapter discussed database mail and a sprinkling of business intelligence/analysis-oriented features related to database management.