CHAPTER 1

Administering SQL Server 2005 Database Engine

Although SQL Server 2005 is composed of numerous components, one component is often considered the foundation of the product. The Database Engine is the core service for storing, processing, and securing data for the most challenging data systems. Likewise, it provides the foundation and fundamentals for the majority of the core database administration tasks. As a result of its important role in SQL Server 2005, it is no wonder that the Database Engine is designed to provide a scalable, fast, and highly available platform for data access and other components.

This chapter focuses on administering the Database Engine component and managing the SQL server properties and database properties based on SQL Server 2005 Service Pack 2. Database Engine management tasks are also covered.

Even though the chapter introduces and explains all the management areas within the Database Engine, you are directed to other chapters for additional information. This is a result of the Database Engine component being so large and intricately connected to other features.

What's New for the Database Engine with Service Pack 2

- Upon the launch of SQL Server 2005, the installation of SQL Server 2005 Integration Services (SSIS) was warranted if organizations wanted to run maintenance plans. This has

since changed. Integration Services is no longer required because maintenance plans are now a fully supported feature within the Database Engine.

- Many enhancements have been made to maintenance plans in SQL Server 2005 with SP2 including support for environments with multiple servers, logging on to remote servers, and providing users with multiple schedules. Previously, maintenance plans could be run only on a server-only installation after SSIS was installed.
- A new storage format is introduced with the release of SP2 to increase functionality and minimize disk space. The new format, known as vardecimal, stores decimal and numeric data as variable-length columns.
- Logon triggers are included with SP2. In addition, SQL Server 2005 Enterprise Edition now has a Common Criteria Compliance Enabled option that follows common criteria for evaluating `SP_CONFIGURE`. See Common Criteria Certification in Books Online for more information.
- Supported with SQL Server 2005 SP2 for the first time is the `sqllogship` application, which is responsible for operations involving backup, copy, and restore procedures. In addition, the application performs cleanup jobs for a log shipping configuration.
- Plan cache improvements are part of the Database Engine enhancements with SP2, improving system performance and improving the use of the physical memory readily available to database pages. Plan cache improvements also can return XML query plans with an XML nesting level greater than or equal to 128 by using the new `sys.dm_exec_text_query_plan` table-valued function. This feature is supported in SQL Server 2005 Express Edition SP2.
- SQL Server Management Studio (SSMS) for Relational Engine features the following:
  - The `Table.CheckIdentityValue()` is supported only with the Express Edition of SQL Server 2005 and is involved in generating a schema name for an object name that meets the criteria.
  - The `Column.AddDefaultConstraint()` feature is also supported only with the Express Edition of SQL Server 2005. This feature is responsible for working against table columns for SQL Server 2000 database instances.

Administering SQL Server 2005 Server Properties

The SQL Server Properties dialog box is the main place you, as database administrator, configure server settings specifically tailored toward a SQL Server Database Engine installation.

You can invoke the Server Properties for the Database Engine by following these steps:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. Connect to the Database Engine in Object Explorer.
3. Right-click SQL Server and then select Properties.

The Server Properties dialog box includes eight pages of Database Engine settings that can be viewed, managed, and configured. The eight Server Properties pages include

- General
- Memory
- Processors
- Security
- Connections
- Database Settings
- Advanced
- Permissions

Note

Each SQL Server Properties setting can be easily scripted by clicking the Script button. The Script button is available on each Server Properties page. The Script options available include Script Action to New Query Window, Script Action to a File, Script Action to Clipboard, and Script Action to a Job.

The following sections provide examples and explanations for each page within the SQL Server Properties dialog box.

Administering the General Page

The first Server Properties page, General, includes mostly informational facts pertaining to the SQL Server 2005 installation, as illustrated in Figure 1.1. Here, you can view the following items: SQL Server Name; Product Version

such as Standard, Enterprise, or 64 Bit; Windows Platform such as Windows 2000 or Windows 2003; SQL Server Version Number; Language Settings; Total Memory in the Server; Number of Processors; Root Directory; Server Collation; and whether the installation is clustered.

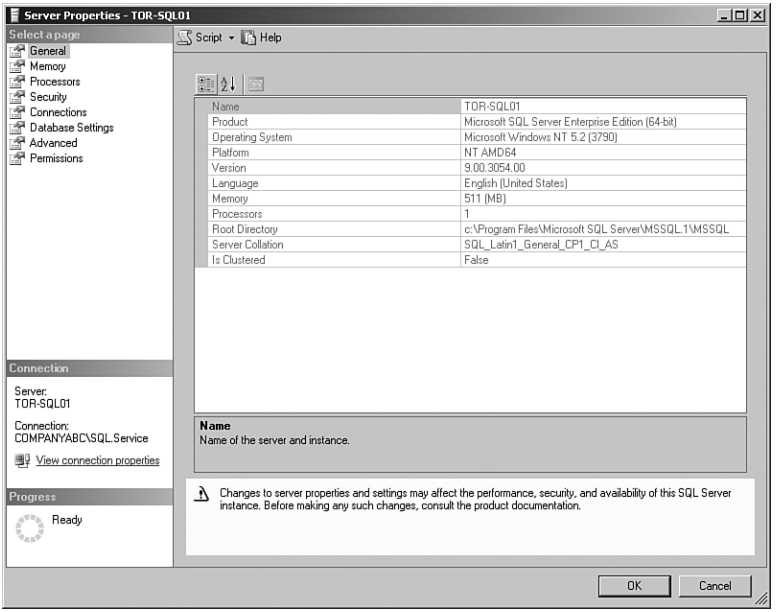

FIGURE 1.1
Administering the Server Properties General page.

Administering the Memory Page

Memory is the second page within the Server Properties dialog box. As shown in Figure 1.2, this page is broken into two sections: Server Memory Options and Other Memory Options. Each section has additional items to configure to manage memory; they are described in the following sections.

Administering the Server Memory Options

The Server Memory options are

- **Use AWE to Allocate Memory**—If this setting is selected, the SQL Server installation leverages Address Windowing Extensions (AWE) memory.

- **Minimum and Maximum Memory**—The next items within Memory Options are for inputting the minimum and maximum amount of memory allocated to a SQL Server instance. The memory settings inputted are calculated in megabytes.

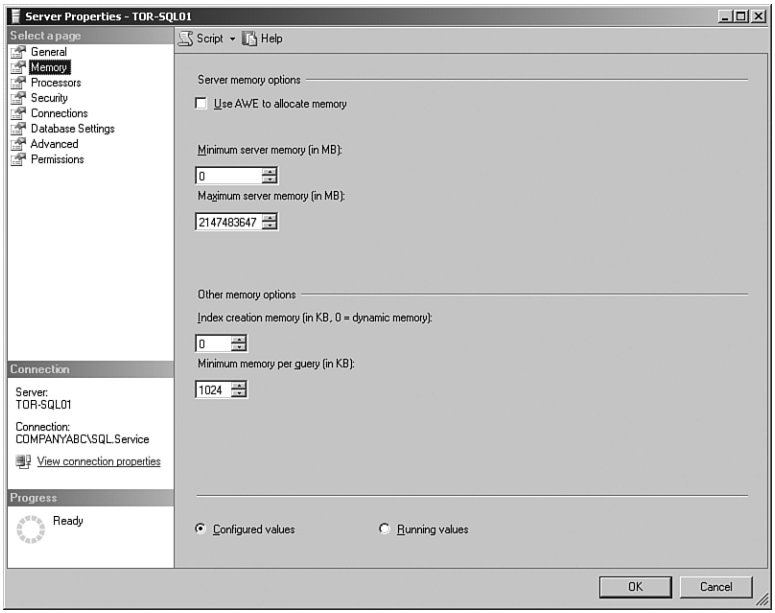

FIGURE 1.2
Administering the Server Properties Memory page.

The following Transact-SQL (TSQL) code can be used to configure Server Memory Options:

```
sp_configure 'awe enabled', 1
RECONFIGURE
GO
sp_configure 'min server memory', "MIN AMOUNT IN MB"
RECONFIGURE
GO
sp_configure 'max server memory', "MAX AMOUNT IN MB"
RECONFIGURE
GO
```

Note

The information in double quotes needs to be replaced with a value specific to this example. This applies to this Transact-SQL example and subsequent ones to follow in this chapter.

Other Memory Options

The second section, Other Memory Options, has two memory settings tailored toward index creation and minimum memory per query:

- **Index Creation Memory**—This setting allocates the amount of memory that should be used during index creation operations.
- **Minimum Memory Per Query**—This setting specifies the minimum amount of memory in kilobytes that should be allocated to a query.

Note

It is best to let SQL Server dynamically manage both the memory associated with index creation and for queries. However, you can specify values for index creation if you're creating many indexes in parallel. You should tweak the minimum memory setting per query if many queries are occurring over multiple connections in a busy environment.

Use the following TSQL statements to configure Other Memory Options:

```
sp_configure 'index create memory, "NUMBER IN KB"
RECONFIGURE
GO
sp_configure 'min memory per query, "NUMBER IN KB"
RECONFIGURE
GO
```

Administering the Processors Page

The Processor page, shown in Figure 1.3, should be used to administer or manage any processor-related options for the SQL Server 2005 Database Engine. Options include threads, processor performance, affinity, and parallel or symmetric processing.

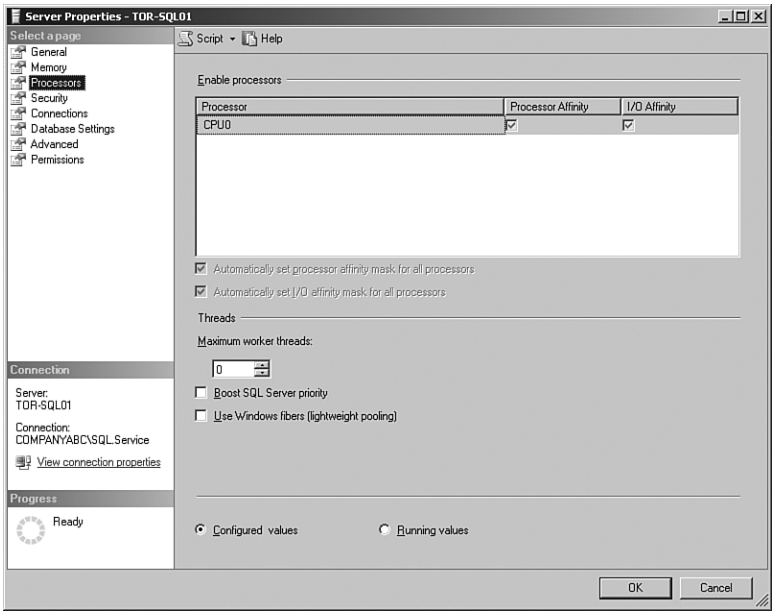

FIGURE 1.3
Administering the Server Properties Processor page.

Enabling Processors

Similar to a database administrator, the operating system is constantly multitasking. Therefore, the operating system moves threads between different processors to maximize processing efficiency. You should use the Processor page to administer or manage any processor-related options such as parallel or symmetric processing. The processor options include

- **Enable Processors**—The two processor options within this section include Processor Affinity and I/O Affinity. Processor Affinity allows SQL Server to manage the processors; therefore, processors are assigned to specific threads during execution. Similar to Processor Affinity, the I/O Affinity setting informs SQL Server on which processors can manage I/O disk operations.

Tip

SQL Server 2005 does a great job of dynamically managing and optimizing processor and I/O affinity settings. If you need to manage these settings manually, you should reserve some processors for threading and others for I/O operations. A processor should not be configured to do both.

- **Automatically Set Processor Affinity Mask for All Processors**—If this option is enabled, SQL Server dynamically manages the Processor Affinity Mask and overwrites the existing Affinity Mask settings.
- **Automatically Set I/O Affinity Mask for All Processors**—Same thing as the preceding option: If this option is enabled, SQL Server dynamically manages the I/O Affinity Mask and overwrites the existing Affinity Mask settings.

Threads

The following Threads items can be individually managed to assist processor performance:

- **Maximum Worker Threads**—The Maximum Worker Threads setting governs the optimization of SQL Server performance by controlling thread pooling. Typically, this setting is adjusted for a server hosting many client connections. By default, this value is set to 0. The 0 value represents dynamic configuration because SQL server determines the number of worker threads to utilize. If this setting will be statically managed, a higher value is recommended for a busy server with a high number of connections. Subsequently, a lower number is recommended for a server that is not being heavily utilized and has a small number of user connections. The values to be entered range from 10 to 32,767.
- **Boost SQL Server Priority**—Preferably, SQL Server should be the only application running on the server; thus, it is recommended to enable this check box. This setting tags the SQL Server threads with a higher priority value of 13 instead of the default 7 for better performance. If other applications are running on the server, performance of those applications could degrade if this option is enabled because those threads have a lower priority.
- **Use Windows Fibers (Lightweight Pooling)**—This setting offers a means of decreasing the system overhead associated with extreme

context switching seen in symmetric multiprocessing environments. Enabling this option provides better throughput by executing the context switching inline.

Note

Enabling fibers is tricky because it has its advantages and disadvantages on performance. This is derived from how many processors are running on the server. Typically, performance gains occur if the system is running a lot of CPUs, such as more than 16; whereas performance may decrease if there are only 1 or 2 processors. To ensure the new settings are optimized, it is a best practice to monitor performance counters, after changes are made.

These TSQL statements should be used to set processor settings:

```
sp_configure 'affinity mask', "VALUE";
RECONFIGURE;
GO

sp_configure 'affinity 1/0 mask', :"VALUE";
RECONFIGURE;
GO
sp_configure 'lightweight pooling', "0 or 1";
RECONFIGURE;
GO

sp_configure 'max worker threads', :"INTEGER VALUE";
RECONFIGURE;
GO

sp_configure 'priority boost', "0 or 1";
RECONFIGURE;
GO
```

Administering the Security Page

The Security page, shown in Figure 1.4, maintains server-wide security configuration settings. These SQL Server settings include Server Authentication, Login Auditing, Server Proxy Account, and Options.

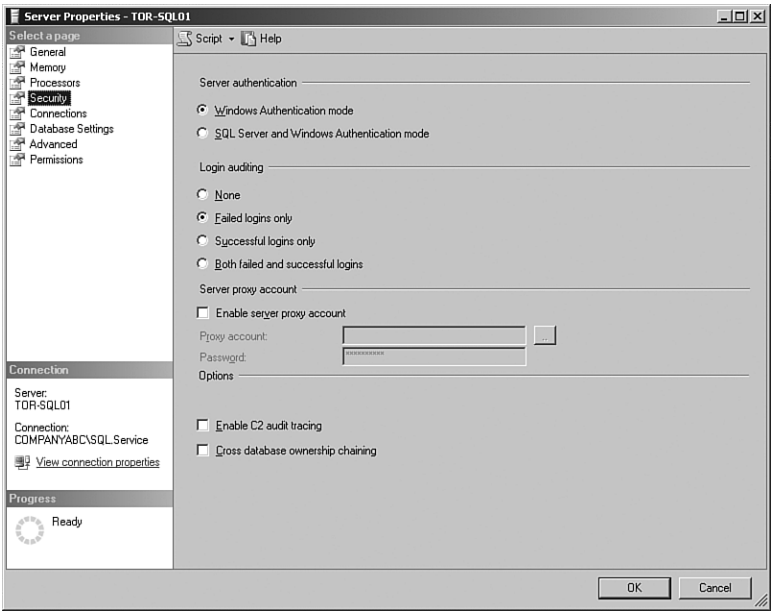

FIGURE 1.4
Administering the Server Properties Security page.

Server Authentication

The first section in the Security page focuses on server authentication. At present, SQL Server 2005 continues to support two modes for validating connections and authenticating access to database resources: Windows Authentication Mode and SQL Server and Windows Authentication Mode. Both of these authentication methods provide access to SQL Server and its resources.

> **Note**
>
> During installation, the default authentication mode is Windows. The authentication mode can be changed after the installation.

The **Windows Authentication Mode** setting is the default Authentication setting and is the recommended authentication mode. It tactfully leverages Active Directory user accounts or groups when granting access to SQL Server. In this mode, you are given the opportunity to grant domain or local

server users access to the database server without creating and managing a separate SQL Server account. Also worth mentioning, when Windows Authentication mode is active, user accounts are subject to enterprise-wide policies enforced by the Active Directory domain, such as complex passwords, password history, account lockouts, minimum password length, maximum password length, and the Kerberos protocol. These enhanced and well-defined policies are always a plus to have in place.

The second Authentication Option is **SQL Server and Windows Authentication (Mixed) Mode.** This setting, which is regularly referred to as *mixed mode authentication*, uses either Active Directory user accounts or SQL Server accounts when validating access to SQL Server. SQL Server 2005 has introduced a means to enforce password and lockout policies for SQL Server login accounts when using SQL Server Authentication. The new SQL Server polices that can be enforced include password complexity, password expiration, and account lockouts. This functionality was not available in SQL Server 2000 and was a major security concern for most organizations and database administrators. Essentially, this security concern played a role in helping define Windows authentication as the recommended practice for managing authentication in the past. Today, SQL Server and Windows Authentication mode may be able to successfully compete with Windows Authentication mode.

> **Note**
>
> Review the authentication sections in Chapter 12, "Hardening a SQL Server 2005 Environment," for more information on authentication modes and which mode should be used as a best practice.

Login Auditing

Login Auditing is the focal point on the second section on the Security page. You can choose from one of the four Login Auditing options available: None, Failed Logins Only, Successful Logins Only, and Both Failed and Successful Logins.

> **Tip**
>
> When you're configuring auditing, it is a best practice to configure auditing to capture both failed and successful logins. Therefore, in the case of a system breach or an audit, you have all the logins captured in an audit file. The drawback to this option is that the log file will grow quickly and will require adequate disk space. If this is not possible, only failed logins should be captured as the bare minimum.

Server Proxy Account

You can enable a server proxy account in the Server Proxy section of the Security page. The proxy account permits the security context to execute operating system commands by the impersonation of logins, server roles, and database roles. If you're using a proxy account, you should configure the account with the least number of privileges to perform the task. This bolsters security and reduces the amount of damage if the account is compromised.

Additional Security Options

Additional security options available in the Options section of the Security page are

- **Enable Common Criteria Compliance**—When this setting is enabled, it manages database security. Specifically, it manages features such as Residual Information Protection (RIP), controls access to login statistics, and enforces restrictions where, for example, the column titled GRANT cannot override the table titled DENY.

> **Note**
>
> Enable Common Criteria Compliance is a new feature associated with SQL Server 2005 Service Pack 2 Enterprise Edition.

- **Enable C2 Audit Tracing**—When this setting is enabled, SQL Server allows the largest number of the success and failure objects to be audited. The drawback to capturing for audit data is that it can degrade performance and take up disk space.
- **Cross Database Ownership Chaining**—Enabling this setting allows cross database ownership chaining at a global level for all databases. Cross database ownership chaining governs whether the database can be accessed by external resources. As a result, this setting should be enabled only when the situation is closely managed because several serious security holes would be opened.

Administering the Connections Page

The Connections page, as shown in Figure 1.5, is the place where you examine and configure any SQL Server settings relevant to connections. The Connections page is broken up into two sections: Connections and Remote Server Connections.

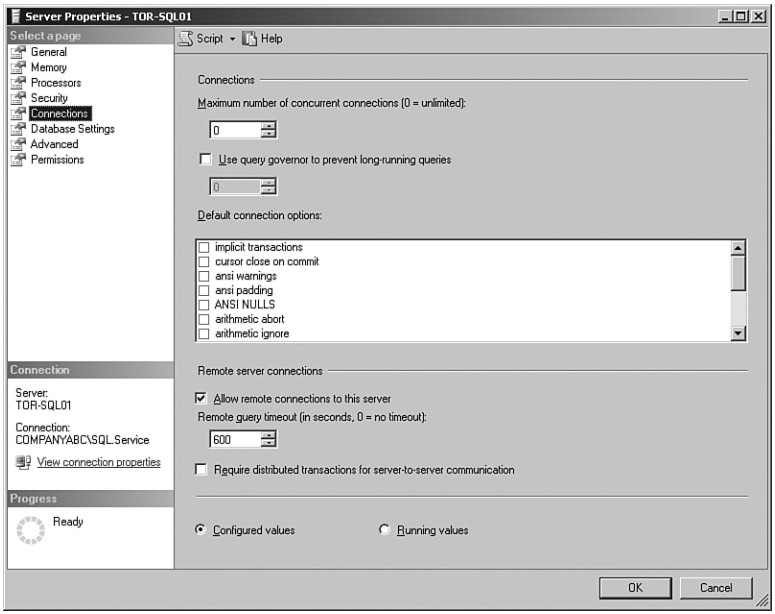

FIGURE 1.5
Administering the Server Properties Connections page.

Connections

The Connections section includes the following settings:

- **Maximum Number of Concurrent Connections**—The first setting determines the maximum number of concurrent connections allowed to the SQL Server Database Engine. The default value is 0, which represents an unlimited number of connections. The value used when configuring this setting is really dictated by the SQL Server hardware such as the processor, RAM, and disk speed.
- **Use Query Governor to Prevent Long-Running Queries**—This setting creates a stipulation based on an upper limit criteria specified on the time period in which a query can run.
- **Default Connection Options**—For the final setting, you can choose from approximately 16 advanced connection options that can be either enabled or disabled, as shown in Figure 1.5.

> **Note**
>
> For more information on each of the default Connection Option settings, refer to SQL Server 2005 Books Online. Search for the topic "Server Properties Connections Page."

Remote Server Connections

The second section located on the Connections page focuses on Remote Server settings:

- **Allow Remote Connections to This Server**— If enabled, the first option allows remote connections to the specified SQL Server.
- **Remote Query Timeout**—The second setting is available only if Allow Remote Connections is enabled. This setting governs how long it will take for a remote query to terminate. The values that can be configured range from 0 to 2,147,483,647. Zero represents infinite.
- **Require Distributed Transactions for Server-to-Server Communication**—The final setting controls the behavior and protects the transactions between systems by using the Microsoft Distributed Transaction Coordinate (MS DTC).

Administering the Database Settings Page

The Database Settings page, shown in Figure 1.6, contains configuration settings that each database within the SQL Server instance will inherit. The choices available on this page are broken out by Fill Factor, Backup and Restore, Recovery, and Database Default Locations.

Default Index Fill Factor

The Default Index Fill Factor specifies how full SQL Server should configure each page when a new index is created. The default setting is 0, and the ranges are between 0 and 100. The 0 value represents a table with room for growth, whereas a value of 100 represents no space for subsequent insertions without requiring page splits. A table with all reads typically has a higher fill factor, and a table that is meant for heavy inserts typically has a low fill factor. The value 50 is ideal when a table has plenty of reads and writes. This setting is global to all tables within the Database Engine.

For more information on fill factors, refer to Chapter 8, "SQL Server 2005 Maintenance Practices" and Chapter 9, "Managing and Optimizing SQL Server 2005 Indexes."

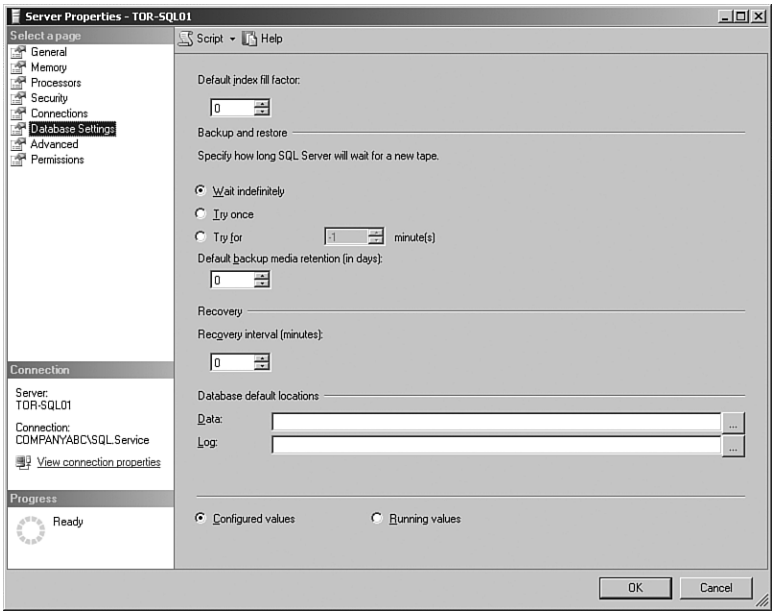

FIGURE 1.6
Administering the Server Properties Database Settings page.

Backup and Restore

The Backup and Restore section of the Database Settings page includes

- **Specify How Long SQL Server Will Wait for a New Tape**—The first setting governs the time interval SQL Server will wait for a new tape during a database backup process. The options available are Wait Indefinitely, Try Once, or Try for a specific number of minutes.
- **Default Backup Media Retention**—This setting is a system-wide configuration that affects all database backups, including the translation logs. You enter values for this setting in days, and it dictates the time to maintain and/or retain each backup medium.

Recovery

The Recovery section of the Database Settings page consists of

- **Recovery Interval (Minutes)**—Only one Recovery setting is available. This setting influences the amount of time, in minutes, SQL

Server will take to recover a database. Recovering a database takes place every time SQL Server is started. Uncommitted transactions are either committed or rolled back.

Database Default Locations

Options available in the Database Default Locations section are

- **Data and Logs**—The two folder paths for Data and Log placement specify the default location for all database data and log files. Click the ellipses on the right side to change the default folder location.

Administering the Advanced Page

The Advanced Page, shown in Figure 1.7, contains the SQL Server general settings that can be configured.

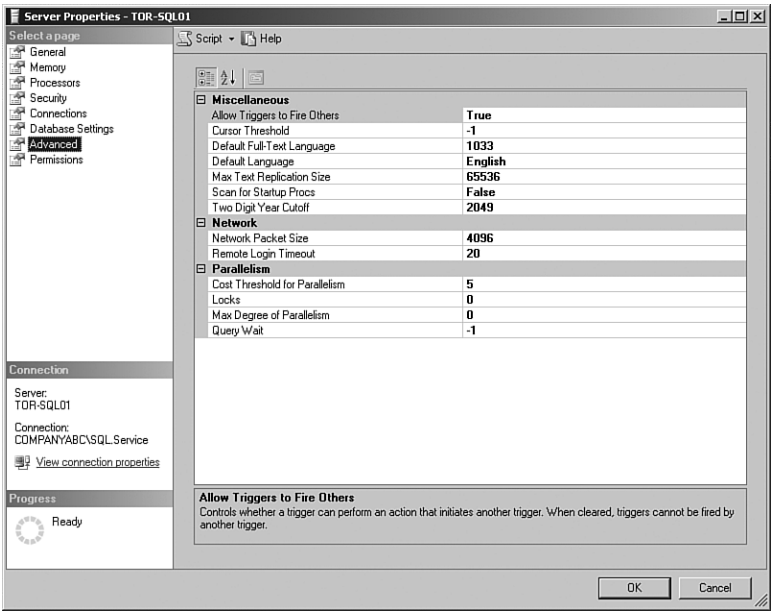

FIGURE 1.7
Administering the Server Properties Advanced Settings page.

Miscellaneous Settings

Options available on the Miscellaneous section of the Advanced page are

- **Allow Triggers to Fire Others**—If this setting is configured to `True`, triggers can execute other triggers. In addition, the nesting level can be up to 32 levels. The values are either `True` or `False`.
- **Cursor Threshold**—This setting dictates the number of rows in the cursor that will be returned for a result set. A value of 0 represents that cursor keysets are generated asynchronously.
- **Default Full-Text Language**—This setting specifies the language to be used for full-text columns. The default language is based on the language specified during the SQL Server instance installation.
- **Default Language**—This setting is also inherited based on the language used during the installation of SQL. The setting controls the default language behavior for new logins.
- **Max Text Replication Size**—This global setting dictates the maximum size of text and image data that can be inserted into columns. The measurement is in bytes.
- **Scan for Startup Procs**—The configuration values are either `True` or `False`. If the setting is configured to `True`, SQL Server allows stored procedures that are configured to run at startup to fire.
- **Two Digit Year Cutoff**—This setting indicates the uppermost year that can be specified as a two-digit year. Additional years must be entered as a four digits.

Network Settings

Options available on the Network section of the Advanced page are

- **Network Packet Size**—This setting dictates the size of packets being transmitted over the network. The default size is 4096 bytes and is sufficient for most SQL Server network operations.
- **Remote Login Timeout**—This setting determines the amount of time SQL Server will wait before timing out a remote login. The default time is 30 seconds, and a value of 0 represents an infinite wait before timing out.

Parallelism Settings

Options available on the Parallelism section of the Advanced page are

- **Cost Threshold for Parallelism**—This setting specifies the threshold above which SQL Server creates and runs parallel plans for queries. The cost refers to an estimated elapsed time in seconds required to run the serial plan on a specific hardware configuration. Set this option only on symmetric multiprocessors. For more information, search for "cost threshold for parallelism option" in SQL Server Books Online.
- **Locks**—The default for this setting is 0, which indicates that SQL Server is dynamically managing locking. Otherwise, you can enter a numeric value that sets the utmost number of locks to occur.
- **Max Degree of Parallelism**—This setting limits the number of processors (up to a maximum of 64) that can be used in a parallel plan execution. The default value of 0 uses all available processors, whereas a value of 1 suppresses parallel plan generation altogether. A number greater than 1 prevents the maximum number of processors from being used by a single query execution. If a value greater than the number of available processors is specified, however, the actual number of available processors is used. For more information, search for "max degree of parallelism option" in SQL Server Books Online.
- **Query Wait**—This setting indicates the time in seconds a query will wait for resources before timing out.

Administering the Permissions Page

The Permissions Page, as shown in Figure 1.8, includes all the authorization logins and permissions for the SQL Server instance. You can create and manage logins and/or roles within the first section. The second portion of this page displays the Explicit permission based on the login or role.

For more information on permissions and authorization to the SQL Server 2005 Database Engine, refer to Chapter 13, "Administering SQL Server Security."

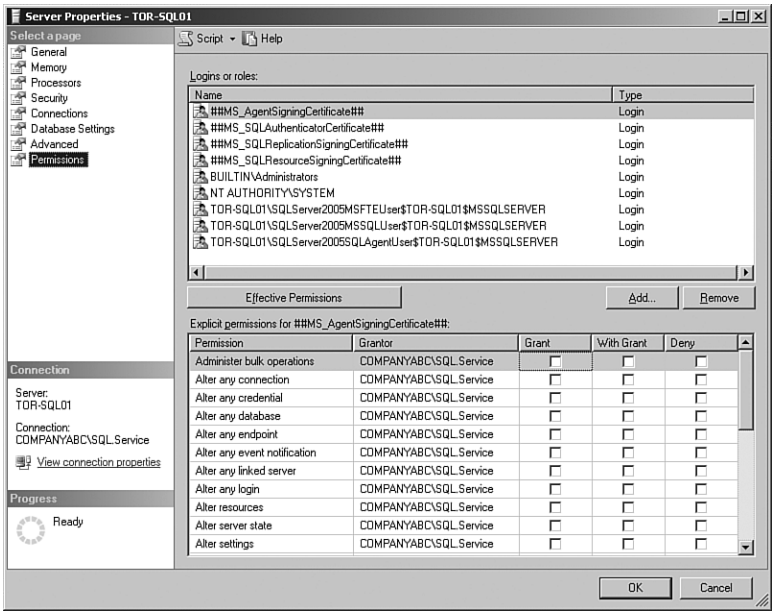

FIGURE 1.8
Administering the Server Properties Permissions page.

Administering the SQL Server Database Engine Folders

After you configure the SQL Server properties, you must manage the SQL Server Database Engine folders and understand what and how the settings should be configured. The SQL Server folders contain an abundant number of configuration settings that need to be managed on an ongoing basis. The main SQL Server Database Engine top-level folders, as shown in Figure 1.9, consist of

- Databases
- Security
- Server Objects
- Replication
- Management
- Notification Services

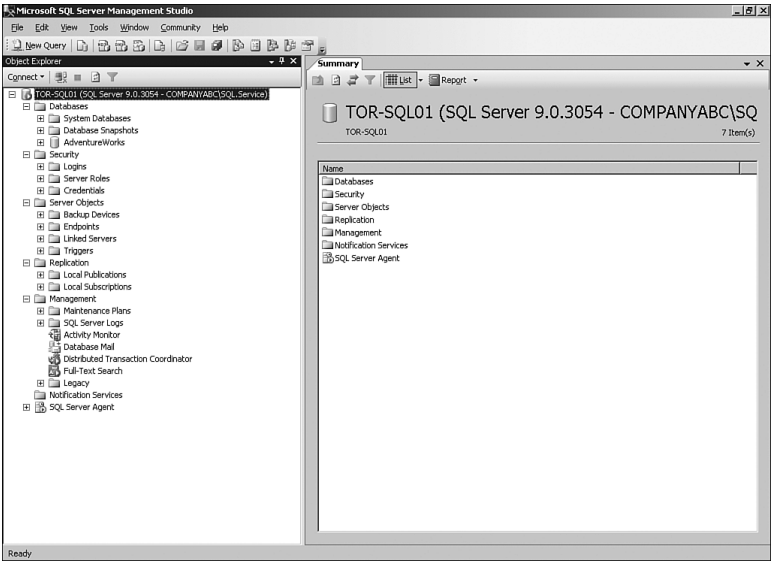

FIGURE 1.9
Viewing the Database Engine folders.

Each folder can be expanded upon, which leads to more subfolders and thus more management of settings. The following sections discuss the folders within the SQL Server tree, starting with the Databases folder.

Administering the Databases Folder

The Databases folder is the main location for administering system and user databases. Management tasks that can be conducted by right-clicking the Database folder consist of creating new databases, attaching databases, restoring databases, and creating custom reports.

The Databases folder contains subfolders as a repository for items such as system databases, database snapshots, and user databases. When a Database folder is expanded, each database has a predefined subfolder structure that includes configuration settings for that specific database. The database structure is as follows: Tables, Views, Synonyms, Programmability, Service Broker, Storage, and Security.

Let's start by examining the top-level folders and then the subfolders in subsequent sections.

Administering the System Databases Subfolder

The System Databases subfolder is the first folder within the Database tree. It consists of all the system databases that make up SQL Server 2005. The system databases consist of

- **Master Database**—The master database is an important system database in SQL Server 2005. It houses all system-level data, including system configuration settings, login information, disk space, stored procedures, linked servers, the existence of other databases, along with other crucial information.
- **Model Database**—The model database serves as a template for creating new databases in SQL Server 2005. The data residing in the model database is commonly applied to a new database with the Create Database command. In addition, the tempdb database is re-created with the help of the model database every time SQL Server 2005 is started.
- **Msdb Database**—Used mostly by the SQL Server Agent, the msdb database stores alerts, scheduled jobs, and operators. In addition, it stores historical information on backups and restores, SQL Mail, and Service Broker.
- **Tempdb—**The tempdb database holds temporary information, including tables, stored procedures, objects, and intermediate result sets. Each time SQL Server is started, the tempdb database starts with a clean copy.

Tip

It is a best practice to conduct regular backups on the system databases. In addition, if you want to increase performance and response times, it is recommended to place the tempdb data and transaction log files on different volumes from the operating system drive. Finally, if you don't need to restore the system databases to a point in failure, you can set all recovery models for the system databases to Simple.

Administering the Database Snapshots Subfolder

The second top-level folder under Databases is Database Snapshots. Database snapshots are a new technology introduced in SQL Server 2005. A *snapshot* allows you to create a point-in-time read-only static view of a database. Typical scenarios for which organizations use snapshots consist of running reporting queries, reverting databases to state when the snapshot was created

in the event of an error, and safeguarding data by creating a snapshot before large bulk inserts occur. All database snapshots are created via TSQL syntax and not the Management Studio.

For more information on creating and restoring a database snapshot, view the database snapshot sections in Chapter 17, "Backing Up and Restoring the SQL Server 2005 Environment" (online).

Administering a User Databases Subfolder

The rest of the subfolders under the top-level Database folder are all the user databases. The user database is a repository for all aspects of an online transaction processing (OLTP) database, including administration, management, and programming. Each user database running within the Database Engine shows up as a separate subfolder. From within the User Database folder, you can conduct the following tasks: backup, restore, take offline, manage database storage, manage properties, manage database authorization, shrink, and configure log shipping or database mirroring. In addition, from within this folder, programmers can create the OLTP database schema, including tables, views, constraints, and stored procedures.

> **Note**
>
> Database development tasks such as creating a new database, views, or stored procedures are beyond the scope of this book, as this book focuses only on administration and management.

Administering the Security Folder

The second top-level folder in the SQL Server instance tree, Security, is a repository for all the Database Engine securable items meant for managing authorization. The sublevel Security Folders consist of

- **Logins**—This subfolder is used for creating and managing access to the SQL Server Database Engine. A login can be created based on a Windows or SQL Server account. In addition, it is possible to configure password policies, server role and user mapping access, and permission settings.
- **Server Roles**—SQL Server 2005 leverages the role-based model for granting authorization to the SQL Server 2005 Database Engine. Predefined SQL Server Roles already exist when SQL Server is deployed. These predefined roles should be leveraged when granting access to SQL Server and databases.

- **Credentials**—Credentials are used when there is a need to provide SQL Server authentication users an identity outside SQL Server. The principal rationale is for creating credentials to execute code in assemblies and for providing SQL Server access to a domain resource.

For more information on the Security folder, authorization, permission management, and step-by-step instructions on how to create logins, server roles, and credentials, refer to Chapter 13.

Administering the Server Objects Folder

The third top-level folder located in Object Explorer is called Server Objects. Here, you create backup devices, endpoints, linked servers, and triggers.

Backup Devices

Backup devices are a component of the backup and restore process when working with OLTP databases. Unlike the earlier versions of SQL Server, backup devices are not needed; however, they provide a great way for managing all the backup data and transaction log files for a database under one file and location.

To create a backup device, follow these steps:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. In Object Explorer, first connect to the Database Engine, expand the desired server, and then expand the Server Objects folder.
3. Right-click the Backup Devices folder and select New Backup Device.
4. In the Backup Device dialog box, specify a Device Name and enter the destination file path, as shown in Figure 1.10. Click OK to complete this task.

This TSQL syntax can be used to create the backup device:

```
USE [master]
GO
EXEC master.dbo.sp_addumpdevice  @devtype = N'disk',
@logicalname = N'Rustom''s Backup Device',
@physicalname = N'C:\Rustom''s Backup Device.bak'
GO
```

For more information on using backup devices and step-by-step instructions on backing up and restoring a SQL Server environment, refer to Chapter 17.

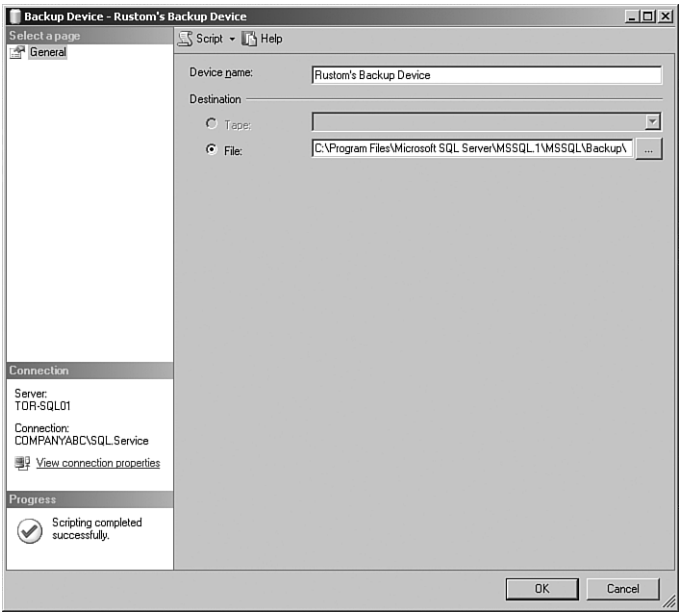

FIGURE 1.10
Creating a backup device with SQL Server Management Studio.

Endpoints

Applications must use a specific port that SQL Server has been configured to listen on to connect to a SQL Server instance. In the past, the authentication process and handshake agreement were challenged by the security industry as not being robust or secure. Therefore, SQL Server 2005 introduces a new concept called *endpoints* to strengthen the communication security process.

The Endpoint folder residing under the Server Objects folder is a repository for all the endpoints created within a SQL Server instance. The endpoints are broken out by system endpoints, database mirroring, service broker, Simple Object Access Protocol (SOAP), and TSQL.

The endpoint creation and specified security options are covered in Chapter 13.

Linked Servers

As the enterprise scales, more and more SQL Server 2005 servers are introduced into an organization's infrastructure. As this occurs, you are challenged by providing a means to allow distributed transactions and queries

between different SQL Server instances. Linked servers provide a way for organizations to overcome these hurdles by providing the means of distributed transactions, remote queries, and remote stored procedure calls between separate SQL Server instances or non–SQL Server sources such as Microsoft Access.

Follow these steps to create a linked server with SQL Server Management Studio (SSMS):

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. In Object Explorer, first connect to the Database Engine, expand the desired server, and then expand the Server Objects Folder.
3. Right-click the Linked Servers folder and select New Linked Server.
4. The New Linked Server dialog box contains three pages of configuration settings: General, Security, and Server Options. On the General Page, specify a linked server name, and select the type of server to connect to. For example, the remote server could be a SQL Server or another data source. For this example, select SQL Server.
5. The next page focuses on security and includes configuration settings for the security context mapping between the local and remote SQL Server instances. On the Security page, first click Add and enter the local login user account to be used. Second, either impersonate the local account, which will pass the username and password to the remote server, or enter a remote user and password.
6. Still within the Security page, enter an option for a security context pertaining to the external login that is not defined in the previous list. The following options are available:
    - **Not Be Made**—Indicates that a login will not be created for user accounts that are not already listed.
    - **Be Made Without a User's Security Context**—Indicates that a connection will be made without using a user's security context for connections.
    - **Be Made Using the Login's Current Security Context—** Indicates that a connection will be made by using the current security context of the user which is logged on.
    - **Be Made Using This Security Context—**Indicates that a connection will be made by providing the login and password security context.

7. On the Server Options page, you can configure additional connection settings. Make any desired server option changes and click OK.

> **Note**
>
> Impersonating the Windows local credentials is the most secure authentication mechanism, provided that the remote server supports Windows authentication.

Triggers

The final folder in the Server Objects tree is Triggers. It is a repository for all the triggers configured within the SQL Server instance. Again, creating triggers is a development task and, therefore, is not covered in this book.

Administering the Replication Folder

Replication is a means of distributing data among SQL Server instances. In addition, replication can also be used as a form of high availability and for offloading reporting queries from a production server to a second instance of SQL Server. When administering and managing replication, you conduct all the replication tasks from within this Replication folder. Tasks include configuring the distributor, creating publications, creating local subscriptions, and launching the Replication Monitor for troubleshooting and monitoring.

Administering, managing, and monitoring replication can be reviewed in Chapter 6, "Administering SQL Server Replication."

Administering the Notification Services Folder

Notification Services is used for developing applications that create and send notifications to subscribers. Because Notification Services is a SQL Server component, Chapter 4, "Administering SQL Server 2005 Notification Services," is dedicated to administering Notification Services.

Administering Database Properties

The Database Properties dialog box is the place where you manage the configuration options and values of a user or system database. You can execute additional tasks from within these pages, such as database mirroring and transaction log shipping. The configuration pages in the Database Properties dialog box include

- General
- Files
- Filegroups
- Options
- Permissions
- Extended Properties
- Mirroring
- Transaction Log Shipping

The upcoming sections describe each page and setting in its entirety. To invoke the Database Properties dialog box, perform the following steps:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. In Object Explorer, first connect to the Database Engine, expand the desired server, and then expand the Databases folder.
3. Select a desired database such as AdventureWorks, right-click, and select Properties. The Database Properties dialog box, including all the pages, is displayed in the left pane.

Administering the Database Properties General Page

General, the first page in the Database Properties dialog box, displays information exclusive to backups, database settings, and collation settings. Specific information displayed includes

- Last Database Backup
- Last Database Log Backup
- Database Name
- State of the Database Status
- Database Owner
- Date Database Was Created
- Size of the Database
- Space Available
- Number of Users Currently Connected to the Database
- Collation Settings

You should use this page for obtaining factual information about a database, as displayed in Figure 1.11.

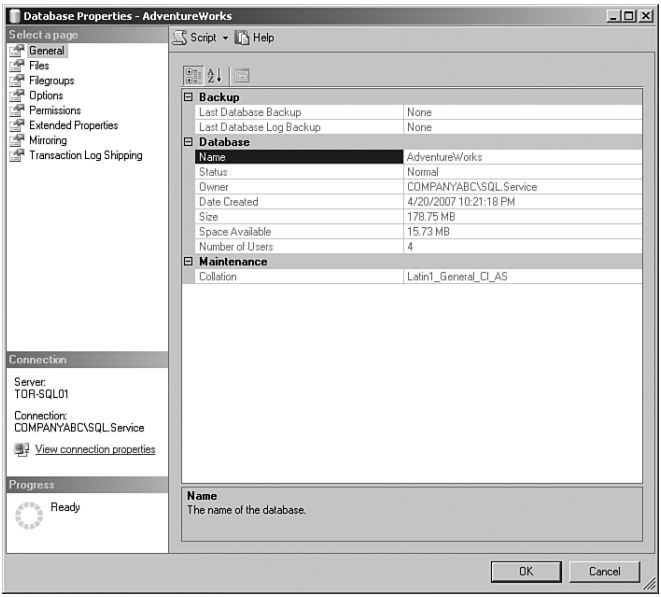

FIGURE 1.11
Viewing the General page in the Database Properties dialog box.

Administering the Database Properties Files Page

The second Database Properties page is called Files. Here, you can change the owner of the database, enable full-text indexing, and manage the database files, as shown in Figure 1.12.

Managing Database Files

The Files page is used to configure settings pertaining to database files and transation logs. You will spend time working in the Files page when initially rolling out a database and conducting capacity planning. Following are the settings you'll see:

- **Data and Log File Types**—A SQL Server 2005 OLTP database is composed of two types of files: data and log. Each database has at least one data file and one log file. When you're scaling a database, it is possible to create more than one data and one log file. If multiple data files exist, the first data file in the database has the extension `*.mdf` and subsequent data files maintain the extension `*.ndf`. In addition, all log files use the extension `*.ldf`.

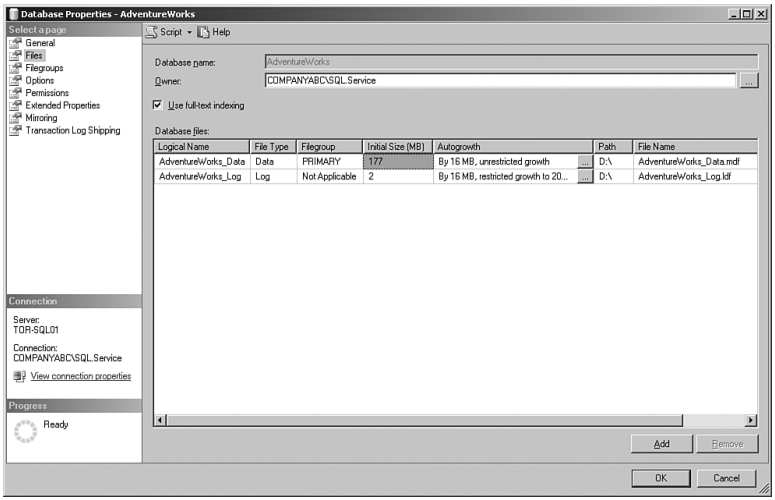

FIGURE 1.12
Configuring the database files settings from within the Files page.

Tip

To reduce disk contention, many database enthusiasts recommend creating multiple data files. The database catalog and system tables should be stored in the primary data file, and all other data, objects, and indexes should be stored in secondary files. In addition, the data files should be spread across multiple disk systems or Logical Unit Number (LUN) to increase I/O performance.

- **Filegroups**—When you're working with multiple data files, it is possible to create filegroups. A filegroup allows you to logically and physically group database objects and files together. The default filegroup, known as the Primary Filegroup, maintains all the system tables and data files not assigned to other filegroups. Subsequent filegroups need to be created and named explicitly.
- **Initial Size in MB**—This setting indicates the preliminary size of a database or transaction log file. You can increase the size of a file by modifying this value to a higher number in megabytes.
- **Autogrowth Feature**—This feature enables you to manage the file growth of both the data and transaction log files. When you click the

ellipses button, a Change Autogrowth dialog box appears. The configurable settings include whether to enable autogrowth, and if autogrowth is selected, whether autogrowth should occur based on a percentage or in a specified number of megabytes. The final setting is whether to choose a maximum file size for each file. The two options available are Restricted File Growth (MB) or Unrestricted File Growth.

Tip

When you're allocating space for the first time to both data files and transaction log files, it is a best practice to conduct capacity planning, estimate the amount of space required for the operation, and allocate a specific amount of disk space from the beginning. It is not a recommended practice to rely on the autogrowth feature because constantly growing and shrinking the files typically leads to excessive fragmentation, including performance degradation.

- **Database Files and RAID Sets**—Database files should reside only on RAID sets to provide fault tolerance and availability, while at the same time increasing performance. If cost is not an issue, data files and transaction logs should be placed on RAID 1+0 volumes. RAID 1+0 provides the best availability and performance because it combines mirroring with stripping. However, if this is not a possibility due to budget, data files should be placed on RAID 5 and transaction logs on RAID 1.

Increasing Initial Size of a Database File

Perform the following steps to increase the data file for the AdventureWorks database using SSMS:

1. In Object Explorer, right-click the AdventureWorks database and select Properties.
2. Select the File Page in the Database Properties dialog box.
3. Enter the new numerical value for the desired file size in the Initial Size (MB) column for a data or log file and click OK.

Creating Additional Filegroups for a Database

Perform the following steps to create a new filegroup and files using the AdventureWorks database with both SSMS and TSQL:

1. In Object Explorer, right-click the AdventureWorks database and select Properties.
2. Select the Filegroups page in the Database Properties dialog box.
3. Click the Add button to create a new filegroup.
4. When a new row appears, enter the name of new the filegroup and enable the option Default.

Alternatively, you can use the following TSQL script to create the new filegroup for the AdventureWorks database:

```
USE [master]
GO
ALTER DATABASE [AdventureWorks] ADD FILEGROUP [SecondFileGroup]
GO
```

Creating New Data Files for a Database and Placing Them in Different Filegroups

Now that you've created a new filegroup, you can create two additional data files for the AdventureWorks database and place them on the newly created filegroup:

1. In Object Explorer, right-click the AdventureWorks database and select Properties.
2. Select the Files page in the Database Properties dialog box.
3. Click the Add button to create new data files.
4. In the Database Files section, enter the following information in the appropriate columns:

Columns	Value
Logical Name	**AdventureWorks_Data2**
File Type	**Data**
FileGroup	**SecondFileGroup**
Size	**10 MB**
Path	**C:**
File Name	**AdventureWorks_Data2.ndf**

5. Click OK.

Note

For simplicity, the file page for the new database file is located in the root of the C: drive for this example. In production environments, however, you should place additional database files on separate volumes to maximize performance.

You can now conduct the same steps by executing the following TSQL syntax to create a new data file:

```
USE [master]
GO
ALTER DATABASE [AdventureWorks]
➥ADD FILE (NAME = N'AdventureWorks_Data2',
➥FILENAME = N'C:\AdventureWorks_Data2.ndf',
➥SIZE = 10240KB , FILEGROWTH = 1024KB )
➥TO FILEGROUP [SecondFileGroup]
GO
```

Configuring Autogrowth on a Database File

Next, to configure autogrowth on the database file, follow these steps:

1. From within the File page on the Database Properties dialog box, click the ellipses button located in the Autogrowth column on a desired database file to configure it.
2. On the Change Autogrowth dialog box, configure the File Growth and Maximum File Size settings and click OK.
3. Click OK on the Database Properties dialog box to complete the task.

You can use the following TSQL syntax to modify the Autogrowth settings for a database file based on a growth rate at 50% and a maximum file size of 1000MB:

```
USE [master]
GO
ALTER DATABASE [AdventureWorks]
MODIFY FILE ( NAME = N'AdventureWorks_Data',
MAXSIZE = 1024000KB , FILEGROWTH = 50%)
GO
```

Administering the Database Properties Filegroups Page

As stated previously, filegroups are a great way to organize data objects, address performance issues, and minimize backup times. The Filegroup page is best used for viewing existing filegroups, creating new ones, marking filegroups as read-only, and configuring which filegroup will be the default.

To improve performance, you can create subsequent filegroups and place data and indexes onto them. In addition, if there isn't enough physical storage available on a volume, you can create a new filegroup and physically place all files on a different volume or LUN if Storage Area Network (SAN) is being used.

Finally, if a database has static data, it is possible to move this data to a specified filegroup and mark this filegroup as read-only. This minimizes backup times; because the data does not change, SQL Server marks this file group and skips it.

> **Note**
>
> Alternatively, you can create a new filegroup directly in the Files page by adding a new data file and selecting New Filegroup from the Filegroup drop-down list.

Administering the Database Properties Options Page

The Options page, shown in Figure 1.13, includes configuration settings on Collation, Recovery Model, and other options such as Automatic, Cursor, and Miscellaneous. The following sections explain these settings.

Collation

The Collation setting located on the Database Properties Options page specifies the policies for how strings of character data are sorted and compared, for a specific database, based on the industry standards of particular languages and locales. Unlike SQL Server collation, the database collation setting can be changed by selecting the appropriate setting from the Collation drop-down box.

Recovery Model

The second setting within the Options page is Recovery Model. This is an important setting because it dictates how much data can be retained, which ultimately affects the outcome of a restore.

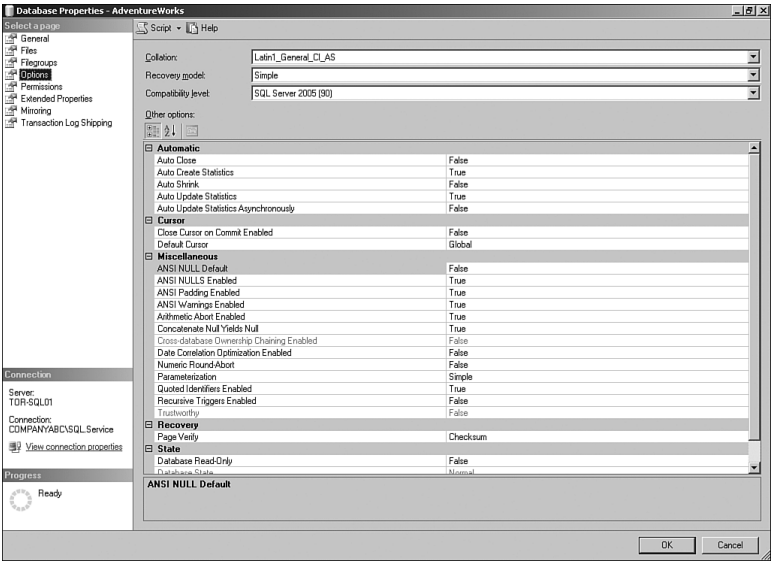

FIGURE 1.13
Viewing and configuring the Database Properties Options page settings.

Understanding and Effectively Using Recovery Models

Each recovery model handles recovery differently. Specifically, each model differs in how it manages logging, which results in whether an organization's database can be recovered to the point of failure. The three recovery models associated with a database in the Database Engine are

- **Full**—This recovery model captures and logs all transactions, making it possible to restore a database to a determined point-in-time or up-to-the-minute. Based on this model, you must conduct maintenance on the transaction log to prevent logs from growing too large and disks becoming full. When you perform backups, space is made available once again and can be used until the next planned backup. Organizations may notice that maintaining a transaction log slightly degrades SQL Server performance because all transactions to the database are logged. Organizations that insist on preserving critical data often overlook this issue because they realize that this model offers them the highest level of recovery capabilities.

- **Simple**—This model provides organizations with the least number of options for recovering data. The Simple recovery model truncates the transaction log after each backup. This means a database can be recovered only up until the last successful full or differential database backup. This recovery model also provides the least amount of administration because transaction log backups are not permitted. In addition, data entered into the database after a successful full or differential database backup is unrecoverable. Organizations that store data they do not deem as mission critical may choose to use this model.
- **Bulk-Logged**—This recovery model maintains a transaction log and is similar to the Full recovery model. The main difference is that transaction logging is minimal during bulk operations to maximize database performance and reduce the log size when large amounts of data are inserted into the database. Bulk import operations such as BCP, BULK INSERT, SELECT INTO, CREATE INDEX, ALTER INDEX REBUILD, and DROP INDEX are minimally logged.

  Since the Bulk-Logged recovery model provides only minimal logging of bulk operations, you cannot restore the database to the point of failure if a disaster occurs during a bulk-logged operation. In most situations, an organization will have to restore the database, including the latest transaction log, and rerun the Bulk-Logged operation.

  This model is typically used if organizations need to run large bulk operations that degrade system performance and do not require point-in-time recovery.

Note

When a new database is created, it inherits the recovery settings based on the Model database. The default recovery model is set to Full.

Next, you need to determine which model best suits your organization's needs. The following section is designed to help you choose the appropriate model.

Selecting the Appropriate Recovery Model

It is important to select the appropriate recovery model because doing so affects an organization's ability to recover, manage, and maintain data.

For enterprise production systems, the Full recovery model is the best model for preventing critical data loss and restoring data to a specific point in time. As long as the transaction log is available, it is possible to even get up-to-the-minute recovery and point-in-time restore if the end of the transaction log is backed up and restored. The trade-off for the Full recovery model is its impact on other operations.

Organizations leverage the Simple recovery model if the data backed up is not critical, data is static or does not change often, or if loss is not a concern for the organization. In this situation, the organization loses all transactions since the last full or last differential backup. This model is typical for test environments or production databases that are not mission critical.

Finally, organizations that typically select the Bulk-Logged recovery model have critical data, but logging large amounts of data degrades system performance, or these bulk operations are conducted after hours and do not interfere with normal transaction processing. In addition, there isn't a need for point-in-time or up-to-the-minute restores.

Note

It is possible to switch the recovery model of a production database and switch it back. This would not break the continuity of the log; however, there could be negative ramifications to the restore process. For example, a production database can use the Full recovery model and, immediately before a large data load, the recovery model can be changed to Bulk-Logged to minimize logging and increase performance. The only caveat is that the organization must understand it lost the potential for point-in-time and up-to-the-minute restores during the switch.

Switching the Database Recovery Model with SQL Server Management Studio

To set the recovery model on a SQL Server 2005 database using SSMS, perform the following steps:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. In Object Explorer, first connect to the Database Engine, expand the desired server, and then expand the database folder.
3. Select the desired SQL Server database, right-click on the database, and select Properties.
4. In the Database Properties dialog box, select the Options page.

5. In Recovery Model, select either Full, Bulk-Logged, or Simple from the drop-down list and click OK.

Switching the Database Recovery Model with Transact-SQL

It is possible not only to change the recovery model of a database with SQL Server Management Studio, but also to make changes to the database recovery model using Transact-SQL commands such as `ALTER DATABASE`. You can use the following TSQL syntax to change the recovery model for the AdventureWorks Database from Simple to Full:

```
--Switching the Database Recovery model
Use Master
ALTER DATABASE AdventureWorks SET RECOVERY FULL
GO
```

Compatibility Level

The Compatibility Level setting located on the Database Properties Options page is meant for interoperability and backward compatibility of previous versions of SQL Server. The options available are SQL Server 2005 (90), SQL Server 2000 (80), and SQL Server 7.0 (70).

Other Options (Automatic)

Also available on the Database Properties Options page are these options:

- **Auto Close**—When the last user exits the database, the database is shut down cleanly and resources are freed. The values to be entered are either `True` or `False`.
- **Auto Create Statistics**—This setting specifies whether the database will automatically update statistics to optimize a database. The default setting is `True`, and this value is recommended.
- **Auto Shrink**—Similar to the shrink task, if this setting is set to `True`, SQL Server removes unused space from the database on a periodic basis. For production databases, this setting is not recommended.
- **Auto Update Statistics**—Similar to the Auto Create Statistics settings, this setting automatically updates any out-of-date statistics for the database. The default setting is `True`, and this value is recommended.
- **Auto Update Statistics Asynchronously**—If the statistics are out of date, this setting dictates whether a query should be updated first before being fired.

Other Options (Cursor)

The following options are also available on the Database Properties Options page:

- **Close Cursor on Commit Enabled**—This setting dictates whether cursors should be closed after a transaction is committed. If the value is `True`, cursors are closed when the transaction is committed, and if the value is `False`, cursors remain open. The default value is `False`.
- **Default Cursor**—The values available include Global and Local. The Global setting indicates that the cursor name is global to the connection based on the Declare statement. During the Declare Cursor statement, the Local setting specifies that the cursor name is Local to the stored procedure, trigger, or batch.

Other Options (Miscellaneous)

The following options are also available on the Database Properties Options page:

- **ANSI Null Default**—The value to be entered is either `True` or `False`. When set to `False`, the setting controls the behavior to supersede the default nullability of new columns.
- **ANSI Null Enabled**—This setting controls the behavior of the comparison operators when used with null values. The comparison operators consist of Equals (=) and Not Equal To (<>).
- **ANSI Padding Enabled**—This setting controls whether padding should be enabled or disabled. Padding dictates how the column stores values shorter than the defined size of the column.
- **ANSI Warnings Enabled**—If this option is set to `True`, a warning message is displayed if null values appear in aggregate functions.
- **Arithmetic Abort Enabled**—If this option is set to `True`, an error is returned, and the transaction is rolled back if an overflow or divide-by-zero error occurs. If the value `False` is used, an error is displayed; however, the transaction is not rolled back.
- **Concatenate Null Yields Null**—This setting specifies how null values are concatenated. True indicates that string + NULL returns NULL. When False, the result is string.
- **Cross-Database Ownership Chaining**—Settings include either `True` or `False`. `True` represents that the database allows cross-database

ownership chaining, whereas `False` indicates that this option is disabled.

- **Date Correlation Optimization Enabled**—If this option is set to `True`, SQL Server maintains correlation optimization statistics on the date columns of tables that are joined by a foreign key.
    - **Numeric Round-Abort**—This setting indicates how the database will handle rounding errors.
    - **Parameterization**—This setting controls whether queries are parameterized. The two options available are Simple and Forced. When you use Simple, queries are parameterized based on the default behavior of the database, whereas when you use Forced, all queries are parameterized.
- **Quoted Identifiers Enabled**—This setting determines whether SQL Server keywords can be used as identifiers when enclosed in quotation marks.
- **Recursive Triggers Enabled**—When this setting is enabled by setting the value to `True`, SQL Server allows recursive triggers to be fired.
- **Trustworthy**—This setting allows SQL Server to grant access to the database by the impersonation context. A value of `True` enables this setting.
- **VarDecimal Storage Format Enabled**—When this option is set to `True`, the database is enabled for the VarDecimal storage format, which is a feature available only with Service Pack 2.

Other Options (Recovery)

Also available on the Database Properties Options page is

- **Page Verify**—This option controls how SQL Server will handle incomplete transactions based on disk I/O errors. The available options include Checksum, Torn Page Detection, and None.

Other Options (State)

The following options are available on the Database Properties Options page:

- **Read Only**—Setting the database value to `True` makes the database read-only.

The default syntax for managing the read-only state of a database is

```
ALTER DATABASE database_name
<db_update_option> ::=
  { READ_ONLY | READ_WRITE }
```

- **State**—This field cannot be edited; it informs you of the state of the database. Possible states include Online, Offline, Restoring, Recovering, Recovery Pending, Suspect, and Emergency.

  To change the state of a database with TSQL, use the default syntax:

  ```
  ALTER DATABASE database_name
  <db_state_option> ::=
  { ONLINE | OFFLINE | EMERGENCY }
  ```

- **Restrict Access**—This setting manages which users can connect to the database. Possible values include Multiple, Single, and Restricted. The Multiple setting is the default state, which allows all users and applications to connect to the database. Single user mode is meant for only one user to access the database. This is typically used for emergency administration. The final setting, Restricted, allows only members of the db_owner, dbcreator, or sysadmin accounts to access the database.

  The TSQL code for setting the Restrict Access value is as follows:

  ```
  ALTER DATABASE database_name
  <db_user_access_option> ::=
      { SINGLE_USER | RESTRICTED_USER | MULTI_USER }
  ```

Administering the Database Properties Mirroring Page

Most database administrators believe database mirroring is the paramount new feature included with the release of SQL Server 2005. Database mirroring is also a SQL Server high-availability alternative for increasing availability of a desired database. Database mirroring transmits transaction log records directly from one SQL Server instance to another SQL Server instance. In addition, if the primary SQL Server instance becomes unavailable, the services and clients automatically fail over to the mirrored server. Automatic failover is contingent on the settings and versions used.

The Database Properties Mirroring page is the primary tool for configuring, managing, and monitoring database mirroring for a database. The Mirroring page includes configuration settings for security; mirroring operating mode;

and the principal, mirror, and witness server network addresses. For more information on configuring database mirroring, review Chapter 19, "Administering and Managing Database Mirroring" (online).

Administering the Database Properties Permissions Page

The Database Properties Permissions page is used to administer database authorization and role-based access and to control permissions on the database. Chapter 13 covers these topics in their entirety.

Administering the Database Properties Extended Permissions Page

The Database Properties Extended Permissions page is used for managing extended properties on database objects, such as descriptive text, input masks, and formatting rules. The extended properties can be applied to schema, schema view, or column view.

Administering the Database Properties Transaction Log Shipping Page

The final Database Properties page is Transaction Log Shipping. Transaction log shipping is one of four SQL Server 2005 high-availability alternatives similar to database mirroring. In log shipping, transactions are sent from a primary server to the standby secondary server on an incremental basis. However, unlike with database mirroring, automatic failover is not a supported feature.

The configuration settings located on the Transaction Log Shipping page in the Database Properties dialog box are the primary place for you to configure, manage, and monitor transaction log shipping.

For more information on administering transaction log shipping, including step-by-step installation instructions, review Chapter 20, "Administering and Managing Log Shipping" (online).

SQL Server Database Engine Management Tasks

The following sections cover additional tasks associated with managing the SQL Server Database Engine.

Changing SQL Server Configuration Settings

Presently, most of the configuration settings can be changed from within SQL Server Management Studio. These settings can also be changed using the `SP_CONFIGURE` TSQL command. The syntax to change configuration settings is

```
SP_CONFIGURE ['configuration name'], [configuration setting
➥value]
GO
RECONFIGURE WITH OVERRIDE
GO
```

The *`configuration name`* represents the name of the setting to be changed, and the *`configuration setting value`* is the new value to be changed. Before you can change settings, however, you must use the `SP_CONFIGURE` command. You must enable advanced settings by first executing the following script:

```
SP_CONFIGURE 'show advanced options', 1
GO
RECONFIGURE
GO
```

For a full list of configuration options, see SQL Server 2005 Books Online.

Managing Database Engine Informational Reports

To succeed in today's competitive IT industry, you must be armed with information pertaining to SQL Server 2005. SQL Server 2005 introduces a tremendous number of canned reports that can be opened directly from within SQL Server Management Studio. These reports provide information that allows you to maximize efficiency when conducting administration and management duties.

You can open these canned reports by right-clicking a SQL Server instance in Management Studio and selecting Reports and then Standard Reports. The standard reports include

- Server Dashboard
- Configuration Changes History
- Schema Changes History
- Scheduler Health

- Memory Consumption
- Activity - All Blocking Transactions
- Activity - All Cursors
- Activity - Tip Sessions
- Activity - Dormant Sessions
- Activity - Top Connections
- Top Transactions by Age
- Top Transactions by Blocked Transactions Count
- Top Transactions by Locks Count
- Performance - Batch Execution Statistics
- Performance - Object Execution Statistics
- Performance - Top Queries by Average CPU Time
- Performance - Top Queries by Average IP
- Performance - Top Queries by Total CPU Time
- Performance - Top Queries by Total IP
- Server Broker Statistics
- Transaction Log Shipping Status

The standard report titled Server Dashboard, displayed in Figure 1.14, is a great overall report that provides an overview of a SQL Server instance, including activity and configuration settings.

Detaching and Attaching Databases

Another common task you must conduct is attaching and detaching databases.

Detaching a Database

When a database is detached, it is completely removed from a SQL Server instance; however, the files are still left intact and reside on the file system for later use. Before a database can be detached, all user connections must be terminated; otherwise, this process fails. The detach tool includes the options to automatically drop connections, update statistics, and keep full text catalogs.

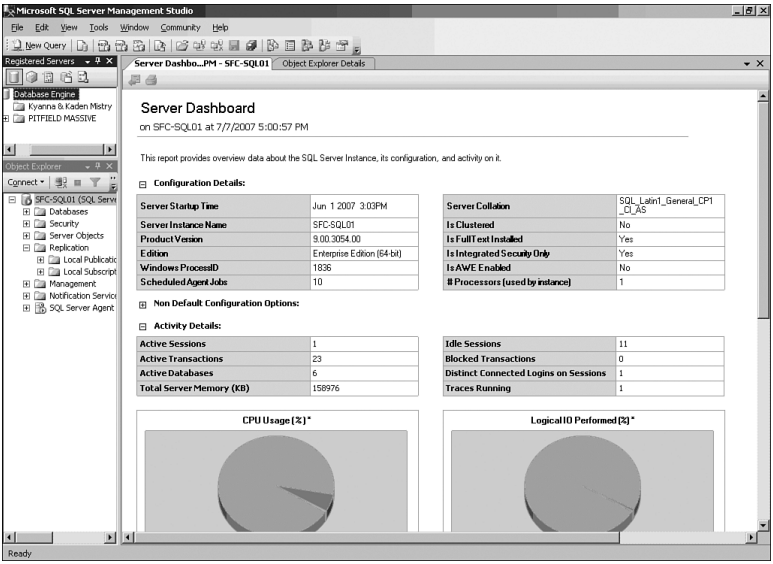

FIGURE 1.14
Viewing the standard Server Dashboard SQL Server canned report.

To drop the sample AdventureWorks database, follow these steps:

1. In Object Explorer, first connect to the Database Engine, expand the desired server, and then expand the Database folder.
2. Select the AdventureWorks database, right-click on the database, select Tasks, and then select Detach.
3. In the Detach Database dialog box, enable the following options, as displayed in Figure 1.15: Drop Connections, Update Statistics, and Keep Full Text Catalogs. Click OK.

Attaching a Database

Here's a common usage scenario for attaching databases: Say you need to move the database from a source to a target SQL Server. When a database is attached, the state of the database is exactly the same as when it was detached.

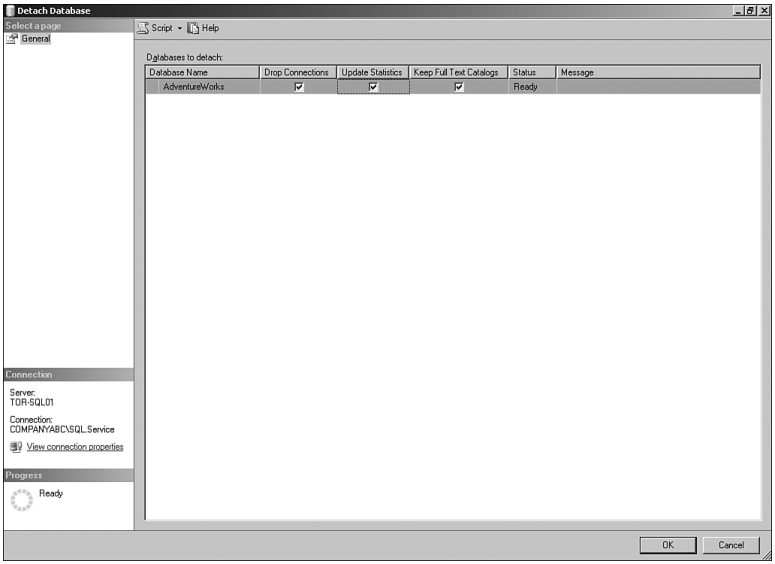

FIGURE 1.15
Specifying detach settings on the Detach Database dialog box.

The following steps illustrate how to attach a database with SQL Server Management Studio:

1. In Object Explorer, first connect to the Database Engine, expand the desired server, and then select the Database folder.
2. Right-click the Database folder and select Attach.
3. In the Attach Databases dialog box, click the Add button to add the database to be attached.
4. In the Locate the Database Files dialog box, specify the path to the `*.mdf` file and click OK.
5. Optionally, change the name or owner of the database.
6. Click OK to attach the database.

Alternatively, you can use the following TSQL syntax to attach the AdventureWorks database:

```
USE [master]
GO
CREATE DATABASE [AdventureWorks] ON
```

```
( FILENAME = N'D:\AdventureWorks_Data.mdf' ),
( FILENAME = N'D:\AdventureWorks_Log.ldf' )
 FOR ATTACH
GO
if exists (select name from master.sys.databases
➥sd where name = N'AdventureWorks' and
➥SUSER_SNAME(sd.owner_sid) = SUSER_SNAME() )
➥EXEC [AdventureWorks].dbo.sp_changedbowner @loginame=
➥N'COMPANYABC\SQL.Service', @map=false
GO
```

Scripting Database Objects

SQL Server 2005 has two levels of scripting functionality that assist you in automatically transforming a SQL Server task or action to a TSQL script. The scripting functionality is a great way to automate redundant administration responsibilities or settings. Moreover, you don't have to be a TSQL scripting expert to create solid scripts.

You can generate a script from within a majority of the SQL Server dialog boxes or pages. For example, if you make changes to the SQL Server Processor Properties page, such as enabling the options Boost SQL Server Priority or User Windows Fibers, you can click the Script button at the top of the screen to convert these changes to a script. In addition, this script can be fired on other SQL Servers to make the configuration automatically consistent across similar SQL Servers.

When you click the Script button, the options available are Script Action to New Query Window, Script Action to File, Script Action to Clipboard, and Script Action to Job.

Another alternative to creating scripts is right-clicking a specific folder within Object Explorer and selecting Script As or right-clicking a database, selecting Tasks, and then selecting Generate Script to invoke the Script Wizard. Some of these tasks include scripting database schemas, jobs, tables, stored procedures, and just about any object within SQL Server Management Studio. Additional scripting statements include Create, Alter, Drop, Select, Insert, and Delete.

Backing Up and Restoring the Database

Creating a backup and recovery strategy is probably the most important task you have on your plate. When you're creating backups, it is imperative that you understand the recovery models associated with each database such as

Full, Simple, and Bulk-Logged and understand the impact of each model on the transaction log and the recovery process. In addition, it is a best practice to back up the user databases, but to restore a full SQL Server environment, the system database should be included the backup strategy.

For more information on recovery models and backing up and restoring a SQL Server environment, see Chapter 17. That chapter focuses on backing up all components of SQL Server, such as the Database Engine, Analysis Services, Reporting Services, and Information Services.

Transferring SQL Server Data

There are many different ways to transfer data or databases from within SQL Server Management Studio. There are tasks associated with importing and exporting data and copying and/or moving a full database with the Copy Database Wizard. To use the transferring tasks, right-click a database, select Tasks, and then select Import Data, Export Data, or Copy Database.

Each of these ways to move data is discussed in its entirety in Chapter 11, "Creating Packages and Transferring Data."

Taking a SQL Server Database Offline

As a database administrator, you may sometimes need to take a database offline. When the database is offline, users, applications, and administrators do not have access to the database until it has been brought back online.

Perform the following steps to take a database offline and then bring it back online:

1. Right-click on a desired database such as AdventureWorks, select Tasks, and then select Take Offline.
2. In the Task Database Offline screen, verify that the status represents that the database has been successfully taken offline and then select Close.

Within Object Explorer, a red arrow pointing downward is displayed on the Database folder, indicating that the database is offline. To bring the database back online, repeat the preceding steps but select Online instead.

In addition, you can use the following TSQL syntax to change the state of a database from Online, Offline, or Emergency:

```
ALTER DATABASE database_name
<db_state_option> ::=
    { ONLINE | OFFLINE | EMERGENCY }
```

Note

When the database option is configured to an Emergency state, the database is considered to be in single-user mode; the database is marked as read-only. This mode is meant for addressing crisis situations.

Shrinking a Database

The Shrink Database task reduces the physical database and log files to a specific size. This operation removes excess space in the database based on a percentage value being entered. In addition, you can enter thresholds in megabytes, indicating the amount of shrinkage that needs to take place when the database reaches a certain size and the amount of free space that must remain after the excess space is removed. Free space can be retained in the database or released back to the operating system.

The following TSQL syntax shrinks the AdventureWorks database, returns freed space to the operating system, and allows for 15% of free space to remain after the shrink:

```
USE [AdventureWorks]
GO
DBCC SHRINKDATABASE(N'AdventureWorks', 15, TRUNCATEONLY)
GO
```

Tip

It is best practice not to select the option to shrink the database. First, when shrinking the database, SQL Server moves pages toward the beginning of the file, allowing the end of the files to be shrunk. This process can increase the transaction log size because all moves are logged. Second, if the database is heavily used and there are many inserts, the database files will have to grow again. SQL 2005 addresses slow autogrowth with instant file initialization; therefore, the growth process is not as slow as it was in the past. However, sometimes autogrow does not catch up with the space requirements, causing performance degradation. Finally, constant shrinking and growing of the database lead to excessive fragmentation. If you need to shrink the database size, you should do it manually when the server is not being heavily utilized.

Alternatively, you can shrink a database by right-clicking a database and selecting Tasks, Shrink, and Database or File.

Renaming a Database

The following steps illustrate how to change the name of a database by using SQL Server Management Studio:

1. In Object Explorer, right-click the name of the database and select Rename.
2. Type in the new name for the database and press Enter.

Administering the SQL Server Agent

The SQL Server Agent is a Microsoft Windows Service that executes scheduled tasks configured as SQL Server jobs. Ultimately, in SQL Server 2005, any task can be transformed into a job; therefore, the task can be scheduled to reduce the amount of time wasted on manual database administration. The SQL Server Agent can be managed from within SQL Server Management Studio.

> **Note**
> The SQL Server Agent service must be running to execute jobs and tasks. This is the first level of defense when you're troubleshooting why agent jobs are not firing.

Administering the SQL Server Agent Properties

Before utilizing the SQL Server Agent, you should first verify and configure the Agent properties to ensure that everything is copacetic.

The SQL Server Agent Properties dialog box has six pages of configuration settings, described in the following sections.

The General Page

The SQL Server Agent page maintains configurable settings such as Auto Restart SQL Server if It Stops Unexpectedly and Auto Restart SQL Server Agent if It Stops Unexpectedly.

From a best practice perspective, both the restart settings should be enabled on mission-critical databases. This prevents downtime in the event of a server outage because the service will restart if failure is inevitable.

You can change the error log path if preferred and configure a send receipt via the `Net send` command. In addition, you can include execution trace

messages to provide meticulous information on SQL Server Agent operations.

The Advanced Page

The Advanced page controls the behavior of SQL Server Event Forwarding and Idle CPU conditions. It is possible to forward unhandled events, all events, or events based on predefined severity levels selected in the drop-down list to a different server. The target server must be specified in the server drop-down list. The differences between unhandled and handled events are that unhandled events forward only events that no alert responds to, whereas handled events forward both the event and the alert. The final section is tailored toward SQL Server Agent and CPU settings. These settings define the conditions when jobs will run based on values such as Average CPU Usage Falls Below in Percentage and And Remains Below This Level for In Seconds.

> **Note**
>
> In enterprise production environments, a SQL Server instance should have enough processing power that these CPU conditions settings are not required.

The Alert System Page

The Alert System page includes all the SQL Server settings for sending messages from agent alerts. The mail session settings are based on the prerequisite task of configuring SQL Server Database Mail. These topics are discussed in Chapter 21, "Monitoring SQL Server 2005" (online).

The Job System Page

The Job System page controls the SQL Server Agent shutdown settings. You can enter a numeric value based on a time increment that governs how long a job can run before automatically being shut down. It is also possible to specify a nonadministrator Job Step Proxy Account to control the security context of the agent; however, this option is available only when you're managing earlier SQL Server Agent versions.

The Connections Page

The Connections Page should be used to configure a SQL Server alias for the SQL Server Agent. An alias is required only if a connection to the Database

Engine will be made without using the default network transport or an alternate named pipe.

The History Page

You should use the final page, History, for configuring the limit size of a job history log setting. The options include setting maximum job history log size in rows and maximum job history rows per job.

Administering SQL Server Agent Jobs

The first subfolder located under the SQL Server Agent is the Job folder. Here, you create new jobs, manage schedules, manage job categories, and view the history of a job.

Follow these steps to create a new job:

1. In Object Explorer, first connect to the Database Engine, expand the desired server, and then expand the SQL Server Agent folder.
2. Right-click the Jobs folder and select New Job.
3. On the General page in the New Job dialog box, enter a name, owner, category, and description for the new job.
4. Ensure that the Enabled check box is set to `True`, as illustrated in Figure 1.16.
5. Click New on the Steps page. When the New Job Steps page is invoked, type a name for the step and enter the type of job this will be. The options range from Transact-SQL, which is the most common, to other items such as stored procedures, Integrations Services packages, and replication. For this example, select TSQL Type and enter the following TSQL syntax in the command window:

```
BACKUP DATABASE [AdventureWorks] TO  DISK =
➥N'C:\Program Files\Microsoft SQL Server
➥\MSSQL.1\MSSQL\Backup\AdventureWorks.bak'
➥WITH NOFORMAT, NOINIT,
➥NAME = N'AdventureWorks-Full Database Backup',
➥SKIP, NOREWIND, NOUNLOAD,  STATS = 10
GO
```

6. From within the General page, parse the command to verify that the syntax is operational and click the Advanced page.

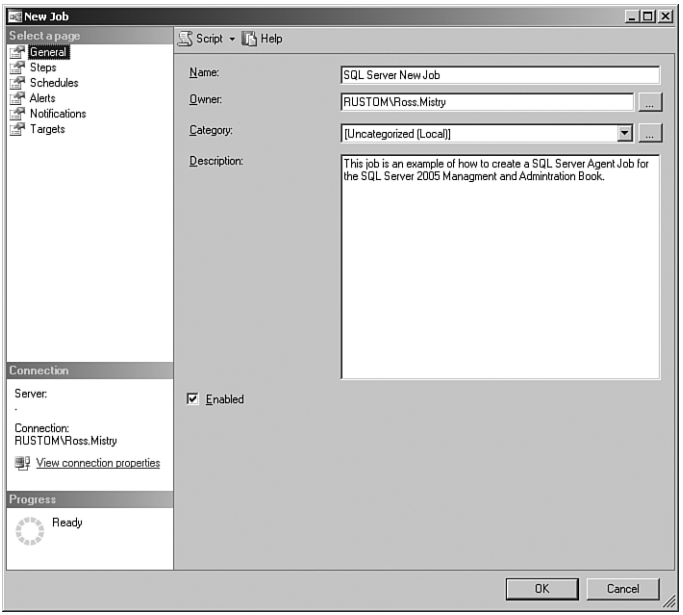

FIGURE 1.16
Specifying the Create New Job Details on the New Job dialog box.

7. The Advanced page includes a set of superior configuration settings. For example, you can specify actions on successful completion of this job, retry attempts including intervals, and what to do if the job fails. This page also includes Output File, Log to Table, History, and the potential to run the job under a different security context. Click OK to continue.
8. Within the New Job dialog box, you can use the Schedules page to view and organize schedules for the job. Here, you can create a new schedule or select one from an existing schedule.
9. Click OK to finalize the creation of the job.

Enabling or Disabling a SQL Server Agent Job

Each SQL Server Agent job can be either enabled or disabled by right-clicking the job and selecting either Enable or Disable.

Viewing SQL Server Agent Job History

From a management perspective, you need to understand whether a SQL Server Agent job was fired properly, completed successfully, or just outright failed. The Job History tool, which is a subcomponent of the Log File Viewer, provides thorough diagnostics and status of job history. Perform the following steps to review job history for a SQL Server Agent job from within SQL Server Management Studio:

1. In Object Explorer, first expand the SQL Server Agent and then the Jobs folder.
2. Right-click a desired job and select View Job History.
3. In the Log File Viewer, review the log file summary for any job from within the center pane.
4. Choose from additional options such as loading saved logs, exporting logs, creating a filter, parsing through logs with the search feature, and deleting logs.

Administering SQL Server Alerts and Operators

The SQL Server Alerts and Operators folders are used for monitoring the SQL Server infrastructure by creating alerts and then sending out notifications to operators. For more information on creating alerts and operators, review Chapter 21.

Administering SQL Server Proxies

The Proxies Folder found within the SQL Server Agent enables you to view or modify the properties of the SQL Server Agent Proxy account. You enter a proxy name and credentials and select the subsystem the proxy account has access to.

Administering SQL Server Error Logs

The final folder in the SQL Server is Error Logs. You can configure the Error Logs folder by right-clicking the folder and selecting Configure. The configuration options include modifying the error log file location, reducing the amount of disk space utilized by enabling the option Write OEM Error Log, and changing the Agent Log Level settings. These settings include enabling Error, Warnings, and/or Information.

Perform the following steps to view SQL Server Agent Error Logs:

1. In Object Explorer, first expand the SQL Server Agent and then the Error Logs folder.
2. When all the error logs are listed under the Error Logs folder, double-click any of the error logs to view them.

Summary

The Database Engine is the core component within SQL Server; it provides a key service for storing, processing, and securing data. SQL Server 2005 Service Pack 2 introduces many new features that improve your success at administering and managing this core component. In addition, reading this chapter will help you to fully understand how to manage and administer the SQL Server instance server properties, Database Engine folders, database properties, and SQL Server Agent.

Best Practices

Following is a summary of some of the best practices from the chapter:

- Leverage the scripting utility within SQL Server Management Studio to transform administration tasks into TSQL code.
- Unless there is a specific need to do otherwise, it is a best practice to allow SQL Server to dynamically manage the minimum and maximum amount of memory allocated to SQL Server. However, if multiple applications are running on SQL Server, it is recommended to specify minimum and maximum values for SQL Server memory. Therefore, the application cannot starve SQL Server by depriving it of memory.
- The preferred authentication mode is Windows Authentication over SQL Server Authentication because it provides a more robust authorization mechanism.
- Configuring SQL auditing is recommended to capture both failed and successful logins.
- Do not set the database to automatically shrink on a regular basis because this leads to performance degradation and excessive fragmentation over time.
- The first Database Engine administration task after a successful SQL installation should involve tuning and configuring the server properties.

- Configure the recovery model for each database accordingly and implement a backup and restore strategy. This should also include the system databases.
- Database files, transaction log files, and operating system files should be located on separate volumes for performance and availability.
- When multiple database files and transaction log files exist, organize them through the use of filegroups.
- Create basic reports in Management Studio to better understand the SQL Server environment.
- Automate administration tasks by using SQL Server 2005 Agent jobs.