CHAPTER 17

Backing Up and Restoring the SQL Server 2005 Environment

Although the key to implementing database technologies is installing the software in a production environment, it is equally important to ensure the new technology environment is being correctly backed up. Having a proper plan and backup process lends assurance that a successful recovery process can be initiated if an organization experiences problems with its systems, servers, or sites.

This chapter covers the areas database administrators and organizations need to consider when conducting SQL Server backups and recoveries. Specifically, this chapter highlights the new facts about backup and recovery included in SQL Server 2005 Service Pack 2. A discussion of the items that should be given consideration when you're planning backup and recovery for SQL Server 2005 follows. The chapter then turns its attention to backing up and recovering SQL Server items including the Database Engine, Analysis Service, Reporting Services, and Full-Text Catalogs. The chapter also includes backing up and restoring Internet Information Services (IIS). Other areas discussed include storage architecture, backup methods, recovery models, and snapshots. In its entirety, this chapter aims at offering you a holistic approach to proper planning, implementing, testing, and support that are essential for achieving a properly backed-up SQL Server 2005 environment.

What's New for Backup and Recovery with Service Pack 2

Microsoft has introduced new features pertaining to backup and recovery in SQL Server 2005 Service Pack 2. The majority of these new backup and recovery features can be found when creating backup strategies with maintenance plans. SQL Server 2005 Service Pack 2 builds on these basic sets of features and comes with the following backup enhancements or bug fixes:

- The Maintenance Plan Designer supports the functionality to create independent schedules for each subplan. Multiple Schedules is a highly anticipated feature and can be leveraged to schedule separate schedules for full, differential, and transaction log backups.
- A new feature has been added to the Database Backup maintenance plan task that includes the ability to choose backup expiration options, such as the backup set will expire after a specific date or on a specific date. This feature existed in SQL Server 2000; however, it was not included in the original release of SQL Server 2005. This item has been introduced with the release of Service Pack 2.
- In the past, when you used the Database Backup maintenance plan, it was possible to create a differential and transaction log backup plan on system databases that used the Simple recovery model. This bug has been addressed.
- The Restore Database dialog restores a database when multiple databases are backed up within a single BAK file.
- The Restore Database dialog allows you to edit the backup location path so that nonmapped network drives can be used for restoring backups.
- The Restore Database dialog recognizes European date/time formats when performing point-in-time restore operations.
- The Backup Database dialog allows the use of a null device as the backup destination.
- The Backup Database dialog with Management Studio Express allows the backup of databases on MSDE 2000 database instances.

The Importance of Backups

Understanding how to back up a SQL Server environment still remains a big challenge for many organizations today, even for those organizations that

make an effort to execute a backup strategy. Unfortunately, in many situations, it takes a disaster for them to realize their backup strategy is inappropriate, or a specific SQL Server component was overlooked. This awakening is far too late for those organizations, however, because they could have already experienced a horrific data loss.

Data loss is unacceptable in today's competitive economy. Losing mission-critical data residing in SQL Server can be particularly harmful to an organization's success because it provides key information that ultimately gives an organization its competitive advantage. Statistics show that organizations suffering catastrophic data loss are more susceptible to going out of business. Moreover, regulatory compliances such as the Sarbanes-Oxley Act place tremendous pressure on organizations to be more trustworthy and accountable for their data, especially financial information.

With organizations beginning to understand the value of their data, more and more organizations are also beginning to recognize the backup and recovery operations of a SQL Server 2005 environment are some of the most important SQL Server administrative tasks of database administrators. When you understand all the intricate details that make up SQL Server and the backup options available, you can better develop a database backup and restoration strategy that minimizes or eliminates data loss and corruption in your organization.

Say you're working with a full installation of SQL Server, which includes all its components. With the ability to identify all these parts and understand how they are associated, you can understand that your focus should be not only on backing up databases, but also on the other SQL Server components and items installed such as Analysis Services, Reporting Services, Full-Text Catalogs, and Internet Information Services. You should take all of these components into account to successfully back up and restore a SQL Server environment.

Items to Consider When Creating a Backup and Recovery Plan

The objective of all backups is to restore a system to a known state. This could include a user or system database, Analysis Services multi-dimensional databases, online analytical processing (OLAP) cubes, reports in Reporting Services, or websites in IIS.

Now that you understand the objective for backups, let's spend a few minutes on the necessity for making backups. The most common need for backups is to recover data from accidental deletions due to human error. Other factors

that might call for a recovery of data may include application errors, hardware failure, or the need to recover data to a prior state.

When organizations understand the objective and necessity for backups, they must attend to the business scope of their SQL Server backup and recovery plan. To help identify the scope, an organization needs to ask some of the following questions:

- Has a Service Level Agreement (SLA) already been defined?
- Is the data in each database considered to be mission critical?
- Is there a clear statement of what should and shouldn't be backed up?
- What is the frequency of the backups?
- What standards are acceptable for offsite storage and retrieval?
- What is the escalation path for failed backups?
- What are the decision criteria for overrun jobs?
- Will the backups be password protected?
- How long will data be retained?
- How much data can the organization afford to lose at any given moment?
- What is the availability of resources governing the backup and restore process?
- What is the financial cost of downtime?

After some of these questions are answered, the high-level scope starts to take shape. An organization then needs to address the technical aspects of the backup and recovery plan. Some of the technical questions may include the following:

- What SQL Server database and components should be included in the backup and recovery plan?
- Which database recovery model should be used?
- How often should the backups occur?
- What type of media should be used?
- Which utilities such as Transact-SQL (TSQL), SQL Server Management Studio (SSMS), Analysis Services, or Internet Information Services should be leveraged when creating backups?

Backing Up and Recovering the Database Engine

The Database Engine component is the heart of SQL Server. It is responsible for storing data, databases, stored procedures, security, and many more functions such as Full-Text Search, Database Mail, replication, and high availability. Because the Database Engine is one of the most crucial components of the SQL Server database as a result of the litany of crucial data it holds, it is essential for organizations to create a backup and recovery plan for the Database Engine.

The Storage Architecture

Executing a successful database backup and restore plan begins with understanding the Database Engine storage architecture. This involves having intimate knowledge of how SQL Server leverages database files, filegroups, and transaction logs.

SQL Server 2005 databases have two kinds of files associated with them: database files and transaction log files. A SQL Server database is always made up of at least one data file and one transaction log file. The following sections elaborate on each of these files.

Database Files

The default database files reside within a primary filegroup. A filegroup is a logical structure that enables you to group data files and manage them as a logical unit. The primary filegroup contains the primary data file and any secondary data files not stored in another filegroup. If you want to increase performance and the speed of the backup and recovery, it is recommended that you create additional files or filegroups and split database objects across these several filegroups to establish parallel processing.

The default extension for the database file within the primary filegroup is `.mdf`. Likewise, filegroups inherit the default extension `.ndf`. It is possible to create up to 32,766 user-defined filegroups.

Transaction Log Files

Every relational database has a transaction log to record database activity. Transaction logs are responsible for recording every modification made to the database. As such, it is a critical component of the database, especially during recovery because it is counted on to restore the database to a point in time or the point of failure. The default extension for a transaction log is

`.ldf`. Similar to database files, additional transaction log files can be added to increase performance, backups, and restore times.

When you're confident you understand the database and transaction log files within the Storage Engine, you should turn your attention to the various Database Engine recovery models in SQL Server. The level of understanding you have of each of these models significantly affects your database backup and restore strategy.

Effectively Using Recovery Models

Each 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 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. It 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 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.

Because 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 has 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.

Now that you're familiar with the three recovery models, you need to determine which model best suits your organization's needs. The next section is designed to help you choose the appropriate model.

Selecting the Appropriate Recovery Model

Selecting the appropriate recovery model 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 tail-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 and does not change often, or loss is not a concern. 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

You are able to switch the recovery model of a production database and switch it back. This does 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 your 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 SSMS

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 tab.
5. In the Recovery Model field, select either Full, Bulk-Logged, or Simple from the drop-down list, as shown in Figure 17.1, and click OK.

Switching the Database Recovery Model with TSQL

You can not only change the recovery model of a database with SSMS, but also make changes to the database recovery model using TSQL 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
```

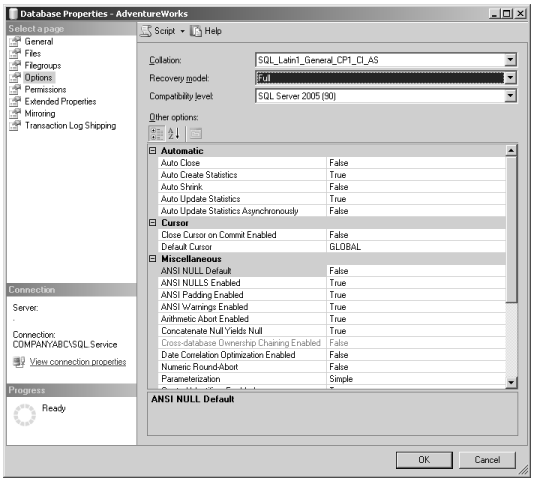

FIGURE 17.1
Selecting a recovery model.

SQL Server Backup Methods

Now that you've selected the appropriate recovery model, you should turn your attention to fully understanding the different backup methods available. This is the third step in successfully creating a backup and recovery solution. The backup utility included in SQL Server offers several options for backing up databases. The following sections identify the following SQL Server backup methods:

- Full backups
- Differential backups
- Transaction log backups
- File and filegroup backups
- Partial backups
- Differential partial backups
- Copy-only backups
- Mirrored backups

Full Backup

The full backup is also commonly referred to as *full database backup*. Its main function is to back up the entire database as well as transaction logs, filegroups, and files. As a result, a full backup can be used to restore the entire database to its original state when the backup was completed.

Equally important, many people refer to the full database backup as the *baseline* for all other backups. The reason is that the full database backup must be

restored before all other backups can be created or restored, such as differential backups, partial backups, and transaction logs.

Listing 17.1 illustrates how to conduct a full database backup for the AdventureWorks database.

> **Note**
>
> For this example and others in this chapter, the backup set is located on the X drive on a proprietary backup file called Rustom. Please change the syntax in these examples to reflect the backup location of your choice based on your environment.

Listing 17.1 **Full Database Backup Sample for AdventureWorks**

```
--FULL DATABASE BACKUP SAMPLE FOR ADVENTUREWORKS
USE AdventureWorks
BACKUP DATABASE [AdventureWorks] TO  DISK
➥ = N'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  DESCRIPTION = N'AdventureWorks-Full Database Backup',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-Full Database Backup',
SKIP,
NOREWIND,
NOUNLOAD,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
declare @backupSetId as int
select @backupSetId = position from msdb..backupset
➥ where database_name=N'AdventureWorks' and
➥ backup_set_id=(select max(backup_set_id)
➥ from msdb..backupset where database_name=N'AdventureWorks' )
if @backupSetId is null begin raiserror(N'Verify failed.
➥ Backup information for database ''AdventureWorks''
➥  not found.', 16, 1) end
RESTORE VERIFYONLY FROM  DISK = N'X:\SQLBackups\
➥ RustomFullBackupExample.bak'
➥ WITH  FILE = @backupSetId,  NOUNLOAD,
NOREWIND
```

Listing 17.1 **continued**

```
GO
--RESULTS
/*
10 percent processed.
20 percent processed.
30 percent processed.
40 percent processed.
50 percent processed.
60 percent processed.
70 percent processed.
80 percent processed.
90 percent processed.
Processed 20976 pages for database 'AdventureWorks',
➥  file 'AdventureWorks_Data' on file 4.
100 percent processed.
Processed 1 pages for database 'AdventureWorks',
➥  file 'AdventureWorks_Log' on file 4.
BACKUP DATABASE successfully processed 20977 pages
➥  in 8.389 seconds (20.484 MB/sec).
The backup set on file 4 is valid.
*/
```

This example conducts a full database backup on the AdventureWorks database. The additional options consist of Backup Set Will Not Expire, Backing Up to Disk, Append to the Existing Backup Set, Verify Backups When Finished, and Perform Checksum Before Writing to Media.

Differential Backups

Unlike a full database backup, a differential database backup backs up only data that changed after the last successful full database backup was conducted, resulting in a smaller backup.

Listing 17.2 illustrates how to conduct a differential database backup for the AdventureWorks database.

Listing 17.2 **Differential Database Backup Sample for AdventureWorks**

```
--DIFFERENTIAL DATABASE BACKUP SAMPLE FOR ADVENTUREWORKS
USE AdventureWorks
BACKUP DATABASE [AdventureWorks] TO
➥ DISK = N'X:\SQLBackups\RustomFullBackupExample.bak'
```

Listing 17.2 **continued**

```
WITH  DIFFERENTIAL , DESCRIPTION = N'AdventureWorks-
➥ Differential Database Backup',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-Differential Database Backup',
SKIP,
NOREWIND,
NOUNLOAD,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
declare @backupSetId as int
select @backupSetId = position from msdb..backupset
➥  where database_name=N'AdventureWorks'
➥ and backup_set_id=(select max(backup_set_id)
➥ from msdb..backupset where database_
➥ name=N'AdventureWorks' )
if @backupSetId is null begin raiserror
➥ (N'Verify failed. Backup information for
➥ database ''AdventureWorks'' not found.', 16, 1) end
RESTORE VERIFYONLY FROM  DISK = N'X:\SQLBackups\
➥ RustomFullBackupExample.bak'
➥ WITH  FILE = @backupSetId,
➥ NOUNLOAD,  NOREWIND
GO
--RESULTS
/*
58 percent processed.
78 percent processed.
97 percent processed.
Processed 40 pages for database 'AdventureWorks',
➥  file 'AdventureWorks_Data' on file 5.
100 percent processed.
Processed 1 pages for database 'AdventureWorks',
➥  file 'AdventureWorks_Log' on file 5.
BACKUP DATABASE WITH DIFFERENTIAL successfully
➥ processed 41 pages in 0.295 seconds (1.136 MB/sec).
The backup set on file 5 is valid.
*/
```

This differential example creates a copy of all the pages in the database modified after the last successful full or differential AdventureWorks database backup. The additional options consist of Backup Set Will Not Expire, Backing Up to Disk, Append to the Existing Backup Set, Verify Backups When Finished, and Perform Checksum Before Writing to Media.

Transaction Log Backup

Transaction log backups are useful only for those databases using a Full or Bulk-Logged recovery model. The transaction log backs up all data as of the last full backup or transaction log backup. As with a differential backup, it is worth remembering that a transaction log backup can be executed only after a full backup has been performed.

Additional options for backing up the transaction log include

- **Truncate the Transaction Log**—If log records were never truncated, they would constantly grow, eventually filling up the hard disk and causing SQL Server to crash. This option is the default transaction log behavior and truncates the inactive portion of the log.
- **Backup the Tail of the Log**—This option is typically used as the first step when restoring SQL Server to a point in failure or point in time. Backing up the tail portion of the log captures the active log that has not been captured by a previous backup before a disaster occurs. This option allows you to recover the database and replay any transactions that have not been committed to the database or included in the backup sets already taken.

Listing 17.3 illustrates how to create a transaction log backup for the AdventureWorks database.

Listing 17.3 **Transaction Log Backup Sample for AdventureWorks**

```
--TRANSACTION LOG BACKUP SAMPLE FOR ADVENTUREWORKS
USE AdventureWorks
BACKUP LOG [AdventureWorks] TO  DISK =
➥ N'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  DESCRIPTION = N'AdventureWorks-Transaction Log  Backup 1',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-Transaction Log  Backup',
SKIP,
NOREWIND,
```

Listing 17.3 **continued**

```
NOUNLOAD,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
declare @backupSetId as int
select @backupSetId = position from
➥ msdb..backupset where database_name=
➥ N'AdventureWorks' and backup_set_
➥ id=(select max(backup_set_id) from
➥ msdb..backupset where database_
➥ name=N'AdventureWorks' )
if @backupSetId is null begin raiserror
➥ (N'Verify failed. Backup information for database
➥ ''AdventureWorks'' not found.', 16, 1) end
RESTORE VERIFYONLY FROM  DISK = N'X:\SQLBackups\
➥ RustomFullBackupExample.bak'
➥ WITH  FILE = @backupSetId,  NOUNLOAD,
NOREWIND
GO
--RESULTS
/*
100 percent processed.
Processed 11 pages for database 'AdventureWorks',
➥ file 'AdventureWorks_Log' on file 7.
BACKUP LOG successfully processed 11
➥ pages in 0.078 seconds (1.089 MB/sec).
The backup set on file 7 is valid.
*/
```

This example conducts a transaction log database backup on the AdventureWorks database. The additional options consist of Backing Up to Disk, Append to the Existing Backup Set, Verify Backups When Finished, and Perform Checksum Before Writing to Media. The transaction log behavior truncates the transaction log when complete.

Listing 17.4 illustrates how to create a transaction log (tail-log) backup for the AdventureWorks database.

Listing 17.4 **Transaction Log (Tail-Log) Backup Sample for AdventureWorks**

```
--TAIL-LOG TRANSACTION LOG BACKUP SAMPLE FOR ADVENTUREWORKS
USE MASTER
BACKUP LOG [AdventureWorks] TO  DISK =
➥ N'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  NO_TRUNCATE ,
DESCRIPTION = N'AdventureWorks-TransactionLogTailBackup ',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-TransactionLogTailBackup ',
SKIP,
NOREWIND,
NOUNLOAD,
NORECOVERY ,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
declare @backupSetId as int
select @backupSetId = position from
➥ msdb..backupset where database_name=
➥ N'AdventureWorks' and backup_set_id=
➥ (select max(backup_set_id) from msdb..backupset
➥  where database_name=N'AdventureWorks' )
if @backupSetId is null begin raiserror(N'Verify failed.
➥ Backup information for database '
➥ 'AdventureWorks'' not found.', 16, 1) end
RESTORE VERIFYONLY FROM  DISK = N'X:\SQLBackups\
➥ RustomFullBackupExample.bak' WITH
➥ FILE = @backupSetId,  NOUNLOAD,
NOREWIND
GO
--RESULTS
/*
100 percent processed.
Processed 1 pages for database 'AdventureWorks',
➥  file 'AdventureWorks_Log' on file 8.
BACKUP LOG successfully processed 1 pages in
➥ 0.250 seconds (0.004 MB/sec).
The backup set on file 8 is valid.
*/
```

This example conducts a backup of the tail of the transaction log on the AdventureWorks database and leaves the database in the restoring state. The additional options consist of Backing Up to Disk, Append to the Existing Backup Set, Verify Backups When Finished, and Perform Checksum Before Writing to Media. The transaction log behavior truncates the transaction log when complete.

> **Note**
>
> It is a best practice to use the master database when performing the tail-log transaction log backup with TSQL.

File and Filegroup Backups

Instead of conducting a full backup, organizations can back up individual files and filegroups. This backup method is often favorable to organizations that just can't consider backing up or restoring their databases because of size and the time required for the task. When you use file and filegroup backups, backing up the transaction log is also necessary because the database must use the Full or Bulk-Logged recovery model.

The basic syntax for creating filegroup backups is as follows:

```
BACKUP DATABASE <Database_Name, sysname, Database_Name>
   FILE = N'<Logical_File_Name_1,sysname,Logical_File_Name_1>',
   FILEGROUP = N'PRIMARY',
   FILE = N'<Logical_File_Name_2, sysname, Logical_File_Name_2>',
   FILEGROUP = N'<Filegroup_1, sysname, Filegroup_1>'
   TO <Backup_Device_Name, sysname, Backup_Device_Name>
GO
```

Partial Backups

Partial backups are a new feature in SQL Server 2005. Primary filegroups and read-write filegroups are always backed up when a partial backup is executed. Any filegroups marked as read-only are skipped to save time and space. Partial backups should not be confused with differential backups. Unlike differential backups, partial backups are best used when read-only filegroups exist and you have chosen not to back up this data because it is static. If you choose to back up a read-only filegroup, this choice must be identified in the Backup command. It is worth mentioning that it is possible to create a partial backup only with TSQL; this functionality is not included in SSMS.

The basic syntax for creating partial and differential backups is as follows:

```
--Creating a Partial Backup
BACKUP DATABASE { database_name | @database_name_var }
 READ_WRITE_FILEGROUPS [ , <read_only_filegroup> [ ,...n ] ]
  TO <backup_device> [ ,...n ]
  [ <MIRROR TO clause> ] [ next-mirror-to ]
  [ WITH { DIFFERENTIAL | <general_WITH_options> [ ,...n ] } ]
[;]
```

Differential Partial Backups

A differential partial backup has many of the features of a differential backup and a partial backup. Only data that has been modified in the primary filegroups and read-write filegroups and not marked as read-only is backed up since the last partial backup. Similar to partial backups, this functionality if not included in SSMS and can be created only with TSQL.

Copy-Only Backups

The capability to make copy-only backups is new in SQL Server 2005. This backup type provides an entire independent backup of a database without affecting the sequence of the backup and restore process.

A common scenario for creating a copy-only backup is when you need to refresh a staging database from production. You can simply create a copy-only backup and restore it to the staging environment without affecting the sequence of the conventional backup or restore process. SSMS does not support copy-only backups. It is possible, however, to create copy-only backups on both the database files and logs.

The basic syntax for creating a copy-only backup for a database file is as follows:

```
BACKUP DATABASE database_name TO
➥ <backup_device> ... WITH COPY_ONLY …
```

The basic syntax for creating a copy-only backup for a transaction log file is as follows:

```
BACKUP LOG database_name TO <backup_device>
➥... WITH COPY_ONLY …
```

Mirrored Backups

Mirrored backups, also called mirrored media sets, yet another new feature in SQL Server 2005, are a large timesaver. Unlike in the past when you were given the arduous task of creating additional backup copies in the event of a media failure, SQL Server 2005 can create a maximum of four mirrors during a backup operation, which increases reliability. Moreover, SQL Server 2005 also ensures the reliability of the media through database and backup checksums. The only shortcoming to mirrored backups is that the media for each mirror must be the same. For instance, if a backup is committed to tape, all mirrors must also be committed to tape.

A mirrored backup is not necessarily a backup type, per se, but an optional clause available when you're creating full, differential, or transaction log backups.

The following TSQL syntax creates a media set called `AdventureWorksMediaSet` using three tape drives as backup devices:

```
BACKUP DATABASE AdventureWorks TO TAPE = '\\.\tape01', TAPE =
➥'\\.\tape02', TAPE = '\\.\tape03'
WITH
    FORMAT,
    MEDIANAME = 'AdventureWorksMediaSet'
```

> **Note**
>
> For a complete listing of TSQL syntax conventions on backups, including the arguments, options, and explanations, see "Backup Transact-SQL" in the SQL Server 2005 Books Online.

Backing Up and Recovering Examples

The following sections focus on SQL Server 2005 backup and restore strategies for databases within the Database Engine. The examples include backing up all user and system databases to disk with a maintenance plan, backing up the AdventureWorks database using the Full recovery model, and restoring the AdventureWorks database to the point of failure.

Understanding the Need to Back Up the System Databases

If you want to restore a SQL Server 2005 installation, it is imperative not only to back up SQL Server user databases such as AdventureWorks, but also the system databases. The main SQL Server 2005 system databases are

- **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 SQL Server Agent, the msdb database stores alerts, scheduled jobs, and operators. In addition, it also stores historical information on backups and restores, 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.

> **Note**
>
> By default, the master, msdb, and tempdb databases use the Simple recovery model, whereas the model database uses the Full recovery model by default.

It is a best practice to include the system database with the existing user database backup strategy. At a minimum, the system databases should be backed up at the time a configuration is added, changed, or removed relative to a database, login, job, or operator.

Conducting a Full Backup Using SSMS

To perform a full SQL database backup on the AdventureWorks database using SSMS, do the following:

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 AdventureWorks database.

4. Right-click on the AdventureWorks database, select Tasks, and then select Backup.
5. On the General page in the Back Up Database window, review the name of the database being backed up and validate that the Backup Type option is set to Full.
6. Type the desired name and description for the backup, and in the Backup Component section, choose Database, as shown in Figure 17.2.

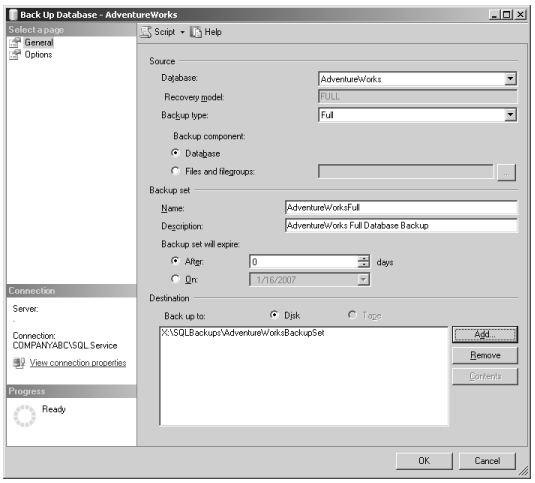

FIGURE 17.2
Viewing the SQL Server Backup screen.

The Destination section identifies the disk or tape media that will contain the backup. You can specify multiple destinations in this section by clicking the Add button. For disk media, a maximum of 64 disk devices can be specified. The same limit applies to tape media. If multiple devices are specified, the backup information is spread across those devices. All the devices must be present to restore the database. If no tape devices are attached to the database server, the Tape option is disabled.

7. In the Destination section, choose the Disk option, as shown in Figure 17.3. Accept the default backup location, or remove the existing path and click Add to select a new destination path for the backup.

8. In Select Backup Destination, type in the path on the hard disk where the database backup will be created, including the backup filename. Click OK. Alternatively, you can choose a backup device instead of storing the backup on hard disk.

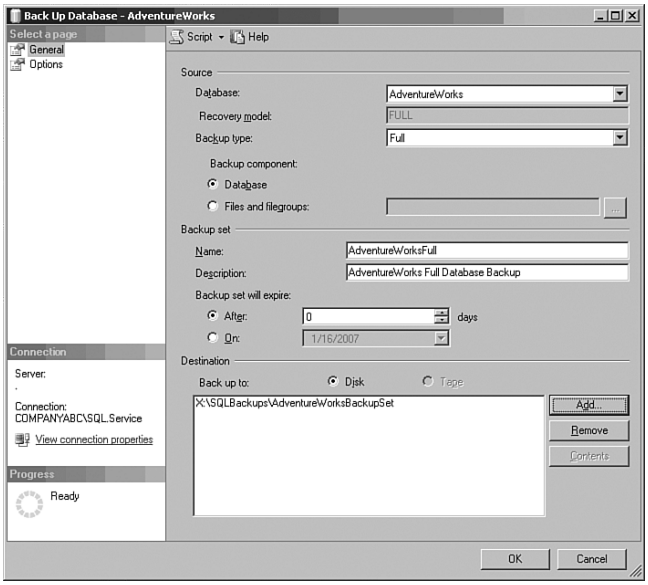

FIGURE 17.3
Providing the database backup file location.

9. Initialize the backup or enter advanced backup options by clicking Options in the Select a Page pane.

   The Overwrite Media section allows you to specify options relative to the destination media for the backup. Keep in mind that a given media set can contain more than one backup. This can occur if the Append to the Existing Backup Set option is selected. With this option, any prior backups that were contained on the media set are preserved and the new backup is added to them. With the Overwrite All Existing Backup Sets option, the media set contains only the latest backup, and all prior backups are not retained.

   Options in the Reliability section can be used to ensure that the backup that has been created can be used reliably in a restore situation.

Verifying the backup when finished is highly recommended but causes the backup time to be extended during the backup verification. Similarly, the Perform Checksum Before Writing to Media option helps ensure that you have a sound backup but again causes the database backup to run longer.

The options in the Transaction Log section are available for databases that are in the Full or Bulk-Logged recovery model. These options are disabled in the Simple recovery model. The Truncate the Transaction Log option causes any inactive portion of the transaction log to be removed after the database backup is complete. This is the default option and helps keep the size of your transaction log manageable. The Backup the Tail of the Log option is related to point-in-time restores.

The last set of options in the Tape Drive section is enabled only when you select Tape for the destination media. The Unload the Tape After Backup option rejects the media tape after the backup is complete. This feature can help identify the end of the backup and prevent the tape from being overwritten the next time the backup runs. The Rewind the Tape Before Unloading option is self-explanatory and causes the tape to be released and rewound prior to unloading the tape.

10. On the Options page, in the Overwrite Media section, maintain the default settings, Back Up to the Existing Media Set and Append to the Existing Backup Set.
11. In the Reliability section, choose the options Verify Backup When Finished, Perform Checksum Before Writing Media, and Continue on Error, as shown in Figure 17.4. Click OK to execute the backup.
12. Review the success or failure error message and click OK to finalize.

Conducting a Differential Backup Using SSMS

To perform a differential SQL database backup on an individual database using SSMS, do the following:

1. Right-click on the AdventureWorks database, select Tasks, and then select Backup.
2. On the General page in the Back Up Database window, review the name of the database being backed up and validate that the Backup Type option is set to Differential.
3. Type the desired name and description for the backup, and in the Backup Component section, choose Database.

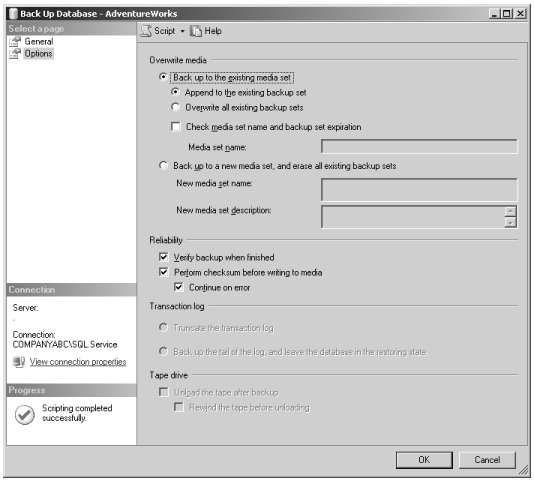

FIGURE 17.4
Setting SQL Server full backup advanced options.

4. In the Destination section, choose the Disk option. Accept the default backup location, or remove the existing path and click Add to select a new destination path for the backup.
5. In Select Backup Destination, type in the path on the hard disk where the database backup will be created, including the backup filename, and then click OK. For this example, use the same destination path and filename used in the previous full backup steps.
6. On the Options page, in the Overwrite Media section, maintain the default settings, Back Up to the Existing Media Set and Append to the Existing Backup Set.
7. In the Reliability section, choose the options Verify Backup When Finished, Perform Checksum Before Writing to Media, and Continue on Error. Then click OK to execute the backup.
8. Review the success or failure error message and click OK to finalize.

Conducting a Transaction Log Backup Using SSMS

To perform a transaction log SQL database backup on an individual database using SSMS, do the following:

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 AdventureWorks database.
4. Right-click on the AdventureWorks database, select Tasks, and then select Backup.
5. On the General page in the Back Up Database window, review the name of the database being backed up and validate that the Backup Type option is set to Transaction Log.
6. Type the desired name and description for the backup, and in the Backup Component section, choose Database.
7. In the Destination section, choose the Disk option. Accept the default backup location, or remove the existing path and click Add to select a new destination path for the backup.
8. In Select Backup Destination, type in the path on the hard disk where the database backup will be created, including the backup filename, and then click OK. For this example, use the same destination path and filename used in the previous full backup steps.
9. Initialize the backup or enter advanced backup options by clicking the Options in the Select a Page pane.

   The Transaction Log section allows you to specify options relative to how the transaction log should be handled during the backup. The two choices are

   - Truncate the Transaction Log
   - Back Up the Tail of the Log, and Leave the Database in the Restoring State

   After a checkpoint is performed, the inactive portion of the transaction log is marked as reusable. If the default option—Truncate the Transaction Log—is selected, the backup truncates the inactive portion of the transaction log, creating free space. The physical size of the transaction log still remains the same, but the usable space is reduced.

   The second option—Back Up the Tail of the Log, and Leave the Database in the Restoring State—is typically used if a disaster occurs and you are restoring the database to a point in failure. Ultimately, this option backs up the active logs that were not already backed up. These active logs can then be used against the recently recovered database to a point in failure or point in time.

10. On the Options page, in the Overwrite Media section, maintain the default settings, Back Up to the Existing Media Set and Append to the Existing Backup Set.
11. In the Reliability section, choose the options Verify Backup When Finished, Perform Checksum Before Writing Media, and Continue on Error. Then click OK to execute the backup.
12. In the Transaction Log section, choose the option Truncate the Transaction Log, as shown in Figure 17.5, and click OK to execute the backup.
13. Review the success or failure error message and click OK to finalize.

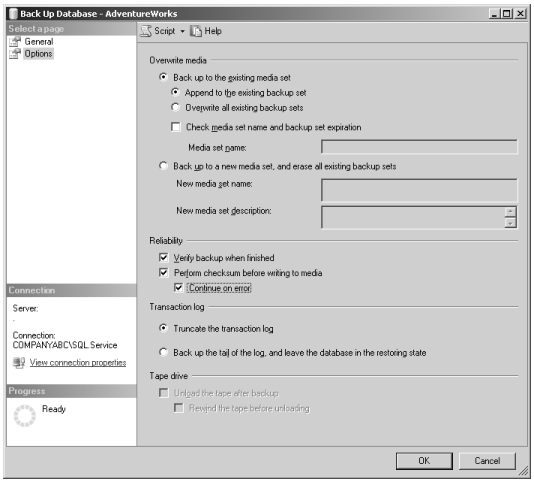

FIGURE 17.5
Setting SQL Server Backup transaction log advanced options.

Automating Backups with a Maintenance Plan

Instead of backing up a database and transaction logs individually with SSMS or TSQL, you can automate and schedule this process with a single task by creating a maintenance plan. The Database Backup maintenance plan reduces the efforts required to create individual backups on all user and system databases. In addition, it is possible to create subtasks and schedule these items at separate times. Maintenance plans are discussed further in Chapter 8, "SQL Server 2005 Maintenance Practices."

Follow these steps to start the creation of a customized backup maintenance plan for all user and system databases by using the Maintenance Plan Wizard:

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 Management folder.
3. Right-click Maintenance Plans and choose Maintenance Plan Wizard.
4. In the Welcome to the Database Maintenance Plan Wizard screen, read the message and click Next.
5. On the Select Plan Properties screen, enter a name and description for the maintenance plan.
6. Choose either the first option, Separate Schedules for Each Task, or the second option, Single Schedule for the Entire Plan or No Schedule. For this example, create a separate schedule for the backup plan, as shown in Figure 17.6. Then click Next.

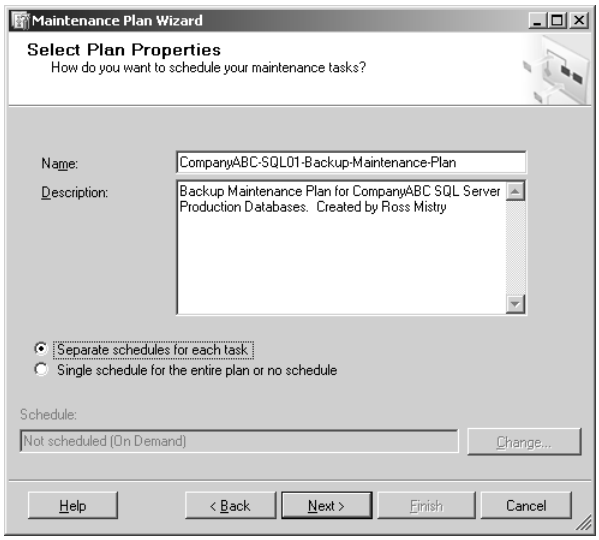

FIGURE 17.6
Selecting the maintenance plan properties.

Note

The ability to create separate independent schedules for each subtask within a single maintenance plan is a new feature supported only with Service Pack 2.

7. In the Select Maintenance Tasks screen, check the Back Up Database (Full) and Back Up Database (Transaction Log) maintenance tasks, as shown in Figure 17.7, and click Next.

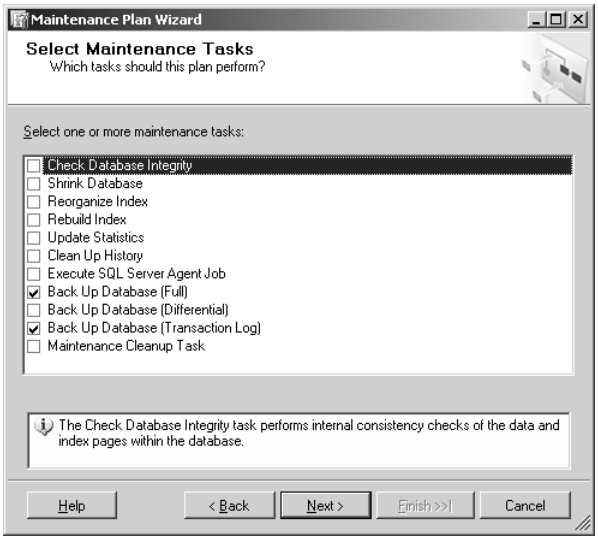

FIGURE 17.7
Selecting maintenance tasks options.

8. On the Select Maintenance Task Order page, review the order in which the tasks will be executed and click Next. For this example, the Back Up Database (Full) task should be listed first and then the Back Up Database (Transaction Log) task should follow.

   The Define Back Up Database (Full) screen includes an expanded set of options when you're creating full backups with the Maintenance Plan Wizard. You can choose a database to be backed up, choose an individual component, set expiration, verify integrity, and decide whether to use disk or tape. The backup options are

- **Specify the Database**—It is possible to generate a maintenance plan to back up an individual database, all databases, systems databases, or all user databases.
- **Backup Component**—In the Backup Component section, you can select either the entire database or individual files or filegroups.
- **Backup Set Will Expire**—This option allows you to specify when the backup set will expire and can be overwritten by another backup based on a number of days or a specific date.
- **Backup Up To**—This option allows the backup to be written to a file or tape. A tape drive must be present on the system, and it is possible to write to a file residing on a network share.
- **Back Up Databases Across One or More Files**—For the backup destination, you can either add or remove one or more disk or tape locations. In addition, you can view the contents of a file and append to the backup file if it already exists.
- **Create a Backup File for Every Database**—Instead of selecting the option Back Up Databases Across One or More Files, you can let SQL Server automatically create a backup file for every database selected. In addition, it is also possible to automatically create a subdirectory for each database selected.

Note

The subdirectory inherits permissions from the parent directory. Therefore, use NTFS permissions to secure this folder and restrict unauthorized access.

- **Verify Backup Integrity**—This option verifies the integrity of the backup when completed by firing a TSQL command that verifies whether the backup was successful and accessible.
- **Schedule**—This option enables you to create a separate schedule for this specific task. This is a new feature included with Service Pack 2.

9. In the Define Back Up Database (Full) Task page, choose All Databases from the drop-down list next to Databases and click OK.
10. Check the Backup Set Will Expire option and enter `0` days so that the backup set will not expire.

Note

With SQL Server 2005 Service Pack 2, the Database Backup maintenance plan task now includes the capability to specify the Backup Expiration option that was previously available in SQL Server 2000.

11. Select the option Create a Backup File for Every Database and then click the ellipses command button to specify a backup folder. In the Locate Folder page, select a folder on the hard disk where the database backup will be created and click OK.
12. Check the option Verify the Backup Integrity.
13. Click Change in the Schedule section. In the Job Schedule Properties page, configure this schedule type to reoccur on a weekly basis. Set the frequency for every Sunday starting at 12:00 a.m., as shown in Figure 17.8, and click OK.

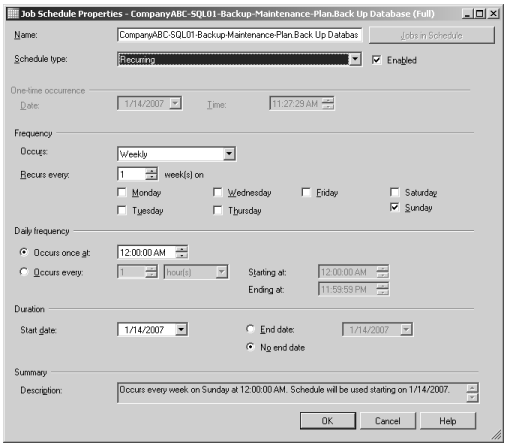

FIGURE 17.8
Setting the maintenance plan job scheduled properties.

14. Review the Define Back Up Database (Full) Task settings, configured as shown in Figure 17.9, and click Next to continue configuring the maintenance plan.

    The next screen, Define Backup Database (Transaction Log) Task, focuses on backup settings for the transaction logs. The settings and options within this screen are similar to the settings and options for

creating full database backups explained previously. The only difference with the transaction logs is that databases using the Simple recovery model are excluded and the backup file extension is `.trn` and not `.bak`.

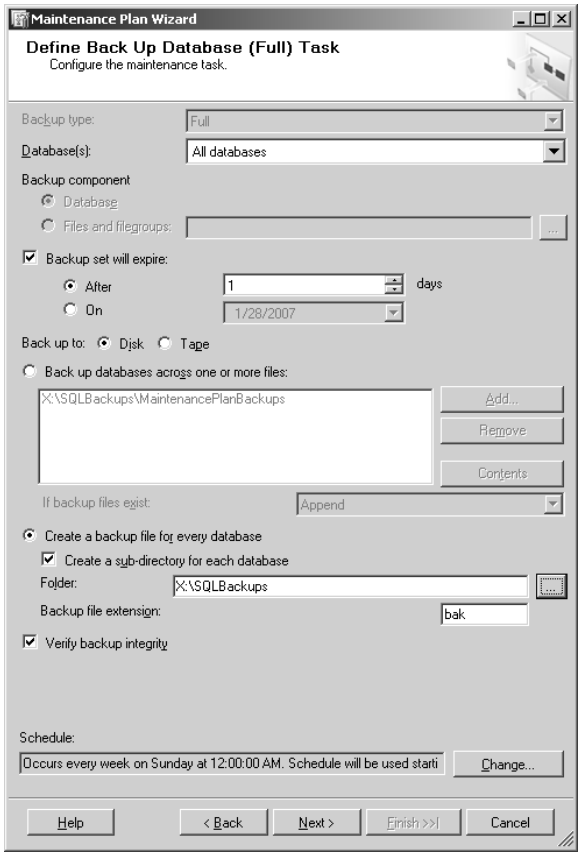

FIGURE 17.9
Selecting the Define Back Up Database (Full) Task settings.

15. Similar to the preceding steps, in the Define Back Up Database (Transaction Log) Task, select All Databases, Backup Set Will Not Expire, Create a Backup File for Every Database, and Create a Sub-Directory for Each Database. In addition, select a backup folder and check the option Verify Backup Integrity, as shown in Figure 17.10.

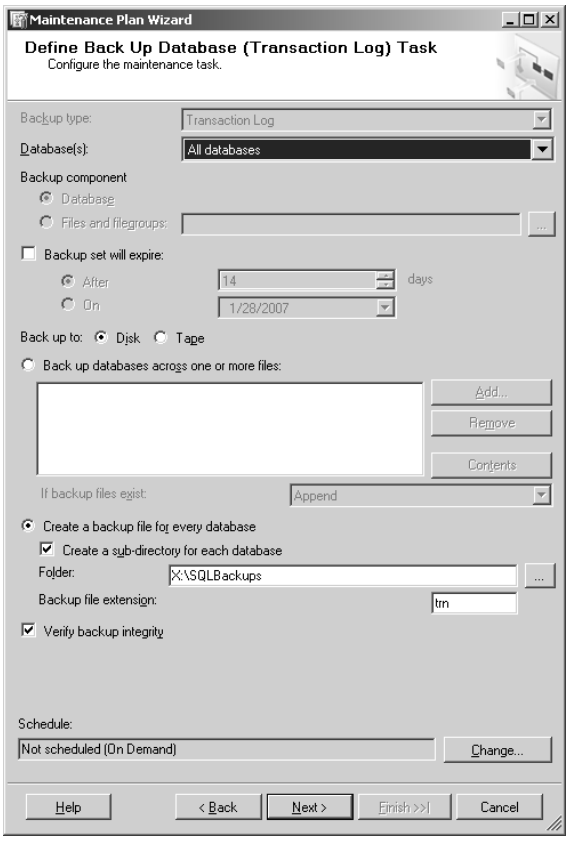

FIGURE 17.10
Selecting the Define Back Up Database (Transaction Log) Task settings.

16. Click Change in the Schedule section. In the Job Schedule Properties page, configure the frequency of this job to occur daily and the daily frequency to occur every 1 hour, as shown in Figure 17.11, and click OK.
17. Review the Define Back Up Database (Transaction Log) Task settings and click Next to continue configuring the maintenance plan.
18. On the Select Report Options page, set the option to either write a report to a text file and enter a folder location, or email the report. If you want to email the report, Database Mail must be enabled and configured, and an Agent Operation with a valid email address must already exist. Click Next to continue.

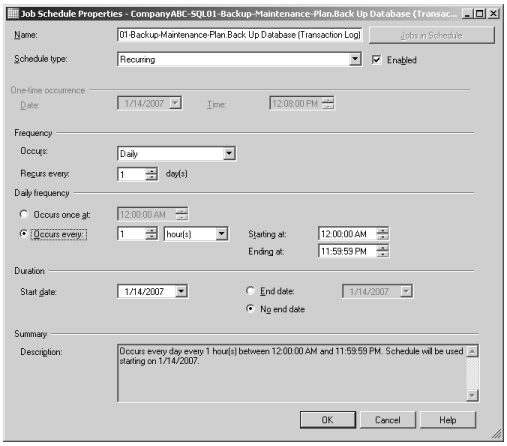

FIGURE 17.11
Selecting maintenance plan job schedule properties settings.

19. The Complete the Wizard page summarizes the options selected in the Maintenance Plan Wizard. It is possible to drill down on a task to view advanced settings. Review the options selected and click Finish to close the summary page.
20. In the Maintenance Plan Wizard Progress screen, review the creation status, as shown in Figure 17.12, and click Close to end the Maintenance Plan Wizard.
21. The maintenance plan is then created and should be visible under the Maintenance Plan folder in SSMS. In addition, you can find the maintenance plan jobs in the Jobs folder within the SQL Server Agent.

Note

For these backup examples, SQL Server is being backed up to disk. In production, backups should not be stored on the same disks as the database or transaction logs. For retention and recovery purposes, backups stored to disks should eventually be committed to tape and stored offsite.

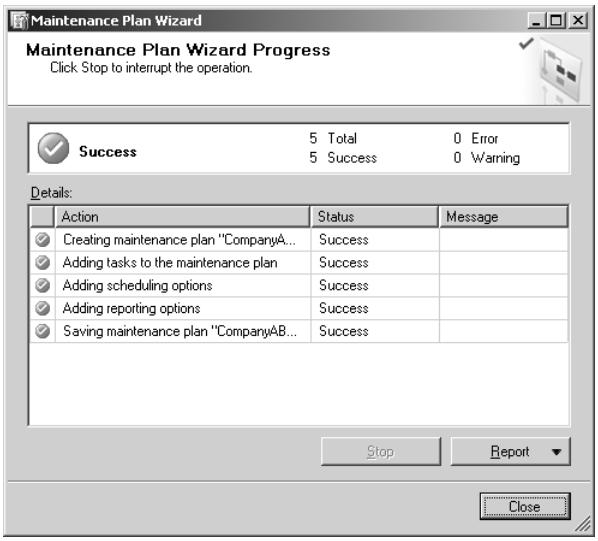

FIGURE 17.12
Viewing the Maintenance Plan Wizard progress.

Conducting a Full Database Recovery Using SSMS

When database corruption or a disaster occurs, you need to restore the database to the point of failure or until a specific date and time. If the database is set to the Full recovery model during the backup, the high-level plan for restoring the database consists of the following sequential tasks: The first step, if possible, is to back up the active transactions (the Tail-Log), and leave the database in a restoring state. The next step includes restoring the latest full and then the most recent differential backups, provided that differential database backups were taken. Finally, subsequent log files should be restored in sequence with the final log being the tail-log.

If the Simple recovery model is used, it is not possible to make transaction log backups; therefore, the restore process consists of restoring the last full backup and most recent differential backup.

Follow these steps to restore the database to the point of failure:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.

2. In Object Explorer, first connect to the Database Engine, click New Query, and execute the syntax in Listing 17.5 to conduct full database, differential, and multiple transaction log backups.

Listing 17.5 **Conducting Full, Differential, and Transaction Log Backups for the AdventureWorks Database**

```
--FULL DATABASE BACKUP SAMPLE FOR ADVENTUREWORKS
USE AdventureWorks
BACKUP DATABASE [AdventureWorks] TO
➥ DISK = N'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  DESCRIPTION = N'AdventureWorks-Full Database Backup',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-Full Database Backup',
SKIP,
NOREWIND,
NOUNLOAD,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
--DIFFERENTIAL DATABASE BACKUP
➥ SAMPLE FOR ADVENTUREWORKS
USE AdventureWorks
BACKUP DATABASE [AdventureWorks] TO
➥ DISK = N'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  DIFFERENTIAL , DESCRIPTION =
➥ N'AdventureWorks-Differential Database Backup',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-Differential Database Backup',
SKIP,
NOREWIND,
NOUNLOAD,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
--TRANSACTION LOG BACKUP SAMPLE FOR ADVENTUREWORKS
USE AdventureWorks
```

Listing 17.5 **continued**

```
BACKUP LOG [AdventureWorks] TO  DISK = N
➥ 'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  DESCRIPTION = N'AdventureWorks-Transaction Log  Backup 1',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-Transaction Log  Backup',
SKIP,
NOREWIND,
NOUNLOAD,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
```

3. To restore the database to the point in failure, first close down any outstanding query windows and then execute the script in Listing 17.6 to perform the tail-log backup to begin disaster recovery.

Listing 17.6 **Performing a Tail-Log Backup to Initiate Disaster Recovery**

```
--TAIL-LOG TRANSACTION LOG BACKUP SAMPLE FOR ADVENTUREWORKS
USE MASTER
BACKUP LOG [AdventureWorks] TO  DISK =
➥ N'X:\SQLBackups\RustomFullBackupExample.bak'
WITH  NO_TRUNCATE ,
DESCRIPTION = N'AdventureWorks-TransactionLogTailBackup ',
NOFORMAT,
NOINIT,
NAME = N'AdventureWorks-TransactionLogTailBackup ',
SKIP,
NOREWIND,
NOUNLOAD,
NORECOVERY ,
STATS = 10,
CHECKSUM,
CONTINUE_AFTER_ERROR
GO
```

Close the query window before moving on to the next step, which includes restoring full, differential, and transaction log backups taken in step 2 using SSMS.

4. In Object Explorer, right-click on the AdventureWorks database, select Tasks, Restore and then select Database. (Notice that the database is in a recovering state and is not operational.)
5. On the General page in the Restore Database window, check all the database, differential, and transaction log backups in the Select the Backup Sets to Restore section, as shown in Figure 17.13. Notice that the tail-log backup is the final backup in the list.

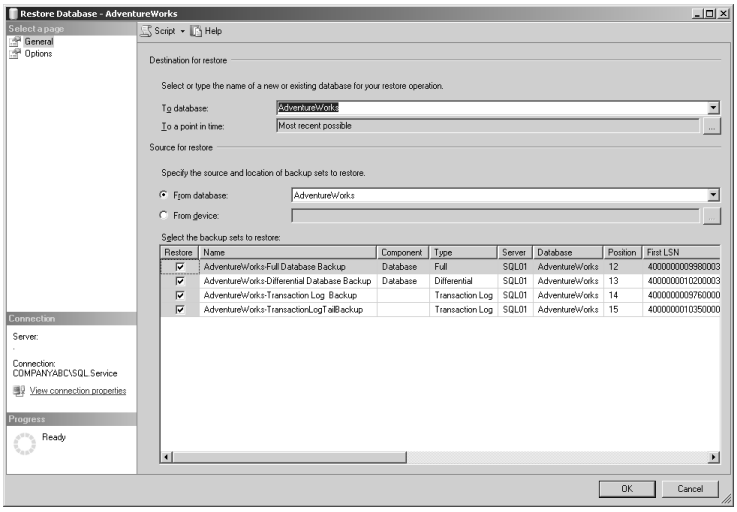

FIGURE 17.13
Selecting general restore options.

6. On the General page in the Restore Database window, select the options Overwrite the Existing Database and Leave the Database Ready to Use by Rolling Back Uncommitted Transactions, as shown in Figure 17.14, and click OK.
7. Review the restore success message and click OK.

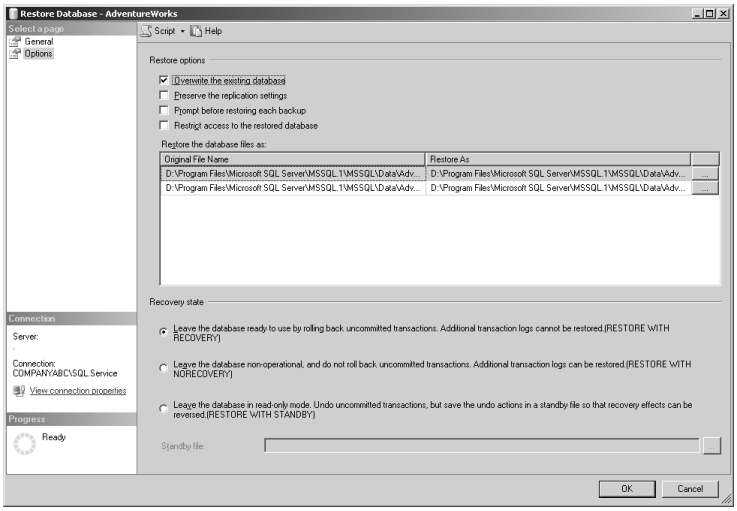

FIGURE 17.14
Selecting additional restore options.

Backing Up and Recovering Reporting Services

A full Reporting Services implementation includes relational reporting databases such as ReportServer and ReportServerTempDB. Many database administrators often make the mistake of overlooking these databases when designing a Reporting Services backup strategy. In truth, if a disaster took place, you would require many other components in addition to these databases to conduct a comprehensive recovery of Reporting Services. The additional components requiring backup attention include report server websites in IIS, encryption keys, custom assemblies, configuration files, and data files.

Backing Up the Report Server Databases

The relational ReportServer and ReportServerTempDB databases need to be backed up. To back up these databases, follow the other database backup steps already described in the chapter. Likewise, you can use a maintenance plan, SQL Server Management Studio, or TSQL to back up the ReportServer and ReportServerTempDB databases.

You can use the following TSQL script to back up the ReportServer and ReportServerTempDB:

```
------------------------------------------------------------
-- Backing Up Whole Database:
➥ ReportServer & ReportServerTempDB
------------------------------------------------------------

USE MASTER
BACKUP DATABASE ReportServer
 TO DISK = 'X:\SQLBackups\ReportServer.bak'
   WITH FORMAT;
GO

USE MASTER
BACKUP DATABASE ReportServerTempDB
 TO DISK = 'X:\SQLBackups\ReportServerTempDB.bak'
   WITH FORMAT;
GO

/*
Results
-----------------------------------------
Processed 336 pages for database 'ReportServer',
➥  file 'ReportServer' on file 1.
Processed 2 pages for database 'ReportServer',
➥  file 'ReportServer_log' on file 1.
BACKUP DATABASE successfully processed 338
➥ pages in 0.514 seconds (5.383 MB/sec).
Processed 176 pages for database 'ReportServerTempDB',
➥  file 'ReportServerTempDB' on file 1.
Processed 2 pages for database 'ReportServerTempDB',
➥  file 'ReportServerTempDB_log' on file 1.
BACKUP DATABASE successfully processed
➥ 178 pages in 0.345 seconds (4.222 MB/sec).
*/
```

The default recovery setting for both the ReportServer and ReportServerTempDB is Simple. For production implementations, however, Microsoft recommends changing the recovery model on the ReportServer database from Simple to Full. This is done so that, in the event of a failure, it is possible to restore the ReportServer database to the point of failure. The ReportServerTempDB's recovery model can maintain the default setting of

Simple. There isn't a need to restore this database to the point of failure because this database is stateless.

The main benefit of backing up the ReportServerTempDB database is to avoid re-creating it in the event of a hardware failure. It is worth noting that if a hardware failure is experienced, you do not need to recover the data in ReportServerTempDB. However, it is the table structure, and not data, that is required.

Backing Up Reporting Services Websites

The Reporting Services websites reside in IIS. Although it depends on the options selected during installation, there is a strong likelihood that the ReportServer is the default website within IIS. With regards to backing up IIS, you should use the IIS backup utility to capture custom IIS settings. It is possible to also encrypt the backup with a password.

Follow these steps to back up Reporting Services websites:

1. Choose Start, All Programs, Administrator Tools, Internet Information Services (IIS) Manager.
2. In IIS Manager, on the server hosting the Reporting Services website, right-click the local computer, point to All Tasks, and click Backup/Restore Configuration.
3. Click Create Backup.
4. In the Configuration Backup Name box, type a name for the backup file.
5. Select the Encrypt Backup Using Password check box, type a password into the Password box, and then type the same password in the Confirm password box.
6. Click OK and then click Close.

Backing Up the Encryption Keys

The encryption keys should be backed up when a Reporting Services installation is configured for the first time. The keys should also be backed up any time the identity of the service accounts is changed or the computer is renamed.

Follow these steps to back up the encryption keys with the Reporting Services Configuration tool:

1. Choose Start, All Programs, Microsoft SQL Server 2005, Configuration Tools, and the Reporting Services Configuration tool.
2. Enter or find the SQL Server name and select the appropriate instance; then click Connect.
3. In the left pane, click Encryption Keys and then click Back Up.
4. In the Encryption Key Information field, type a strong password.
5. Specify a file to contain the stored key. Reporting Services appends an `.snk` file extension to the file. Consider storing the file on a disk so that it is separate from the report server.
6. Click OK and then click Exit.

Tip

To safeguard against recovery and theft, you should *not* store the encryption key on the local server. Instead, you should store it offsite and protect it similarly to other highly privilaged user accounts such as domain administrator passwords and Public Key Infrastructure (PKI) keys.

Backing Up Configuration and Data Files

Reporting Services configuration and data files are the final items that should be included in the backup strategy. The following configuration files should be included in the existing backup policy:

- `Rsreportserver.config`
- `Rswebapplication.config`
- `Rssrvpolicy.config`
- `Rsmgrpolicy.config`
- `Reportingservicesservice.exe.config`
- `Web.config` for both the Report Server and Report Manager ASP.NET applications
- `Machine.config` for ASP.NET

The following Report Designer, Model Designer, and script files should be included in the existing backup policy:

- report definition (.rdl)
- report model (.smdl)
- shared data source (.rds)
- data view (.dv)
- data source (.ds)
- report server project (.rptproj)
- report solution (.sln)
- script files (.rss)

Recovering Reporting Services

To recover Reporting Services, you first need to restore the Reporting Services databases and then restore the encryption key using the Reporting Services Configuration tool. Finally, you can restore additional IIS settings and configuration files.

Backing Up and Recovering Analysis Services

Analysis Services delivers online analytical processing and data mining functionality for business intelligence applications. Analysis Services allows organizations to aggregate data from multiple heterogeneous environments and transform this data into meaningful information that can then be analyzed and leveraged to gain a competitive advantage in the industry. It is equally important to back up Analysis Services databases when using business intelligence solutions in production.

Backing Up Analysis Services Databases

Currently, Analysis Services provides a tool to assist you in manually backing up its databases. The Analysis Services backup file uses the extension `.abf`. As with the databases residing in the Database Engine, however, you need a backup name and location to successfully back up the Analysis Services databases. You also have the opportunity to overwrite backups, apply compression, or encrypt a backup file with a password when backing up the Analysis Services databases.

Unfortunately, it takes a little bit more effort to achieve a successful backup than just choosing the most appropriate backup tool or applying a password to bolster security. To back up the Analysis Services databases correctly and therefore successfully, you must first be aware of the storage models available in SQL Server 2005 Analysis Services because these models affect the backup contents. The three OLAP storage models are

- **Multidimensional OLAP (MOLAP)**—This storage model provides the best query performance because detailed data and aggregations are

kept in a multidimensional compressed proprietary format. The relational source is accessed only during initial processing and queries are resolved entirely from the multidimensional store.

- **Relational OLAP (ROLAP)**—This storage model is considered to be real-time because detailed data and aggregations are stored in the relational source.
- **Hybrid (HOLAP)**—This storage model combines both MOLAP and ROLAP. Aggregations are stored in the multidimensional stores, whereas detailed data is found in the relational store.

> **Note**
>
> For more information on Analysis Services and Storage Models, see Chapter 2, "Administering SQL Server 2005 Analysis Services."

Depending on the storage model you choose, you might need to back up not only the Analysis Services database, but also the relational database as part of the strategy.

Assuming that a storage model has been chosen, you can then follow these steps to back up the Analysis Services databases:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. In Object Explorer, first connect to Analysis Services, expand the desired server, and then expand the database folder.
3. Right-click on the desired database (AdventureWorksDW, for this example) and choose Backup.
4. On the Analysis Services Backup page, next to Backup file, click Browse, type the full path and filename of the backup file to use, and then click OK.
5. Check the Allow File Overwrite, Apply Compression, and Encrypt Backup File check boxes, as shown in Figure 17.15.
6. Enter a strong password into the Password text box and then confirm it by reentering the same password in the Confirm text box. Then click OK.

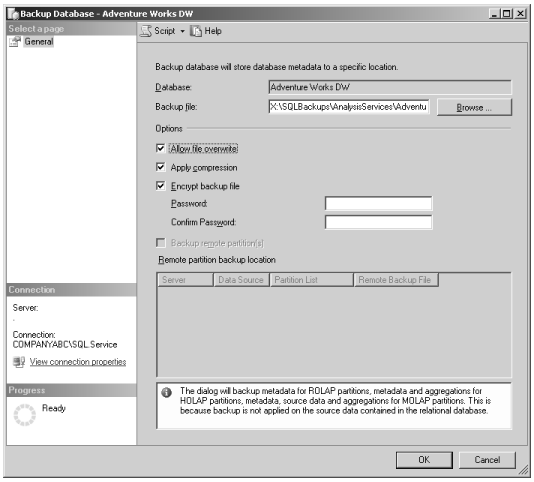

FIGURE 17.15
Selecting Analysis Services backup settings.

Note

Depending on which storage model you choose, do *not* forget to back up the relational databases associated with the OLAP databases.

These steps create the appropriate backup file and can be used during restore purposes. Alternatively, you can use the same backup window to script this task to a job and schedule the job to occur on a reoccurring basis.

Recovering Analysis Services Databases

In the event lost data needs recovering, follow these steps to restore the Analysis Services database:

1. Choose Start, All Programs, Microsoft SQL Server 2005, SQL Server Management Studio.
2. In Object Explorer, first connect to Analysis Services, expand the desired server, and then expand the database folder.
3. Right-click on the desired database (AdventureWorksDW, for this example) and choose Restore.

4. On the Analysis Services Restore Database page, in the Restore Target section, enter the name of the database being restored and type the full path and filename of the backup file to use.
5. In the Options section, select the option Allow Database Overwrite and then enter the password for the backup file, as shown in Figure 17.16, and click OK.

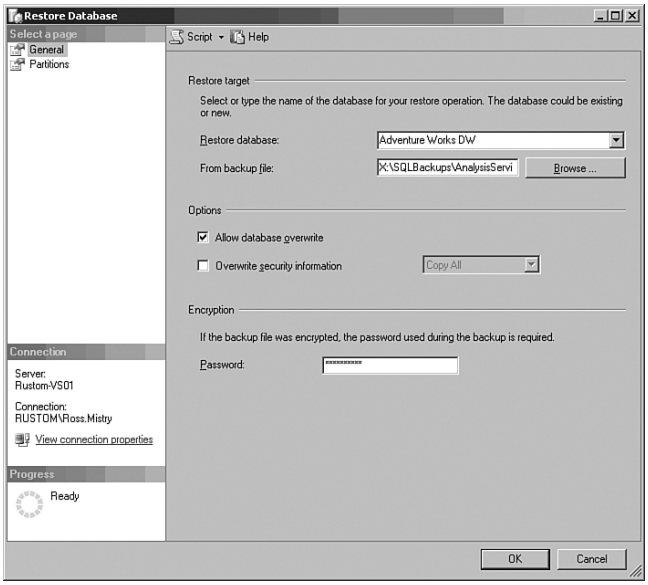

FIGURE 17.16
Selecting additional Analysis Services backup settings.

Backing Up and Recovering Full-Text Catalogs

The process of backing up and restoring full-text catalogs is similar to backing up and restoring database files. Unlike in SQL Server 2000, each full-text catalog in SQL Server 2005 is treated as a file and is automatically included in regular full or differential backups. As a result, when backing up the full-text catalog, you should follow the full and differential backup steps described previously in the chapter.

Backing Up Full-Text Catalogs

Sometimes your organization might want to perform a full or differential backup on just the full-text catalog file and not the actual database. You can use the following syntax to back up the full-text catalog Fulltext_cat and not the database. The backup device should exist prior to executing the script.

```
BACKUP DATABASE database_name
FILE = 'sysft_fulltext_cat'
TO backup_device
```

Restoring Full Text Catalogs

The basic syntax for restoring a full-text catalog is as follows:

```
RESTORE DATABASE database_name
FROM backup_device
```

Understanding and Creating Database Snapshots

Database snapshots are a new feature included with SQL Server 2005. The database snapshot provides a read-only copy of a SQL Server database during a specific point in time. This static view provides a comprehensive picture of the full database, tables, views, and stored procedures. Organizations can use a database snapshot to protect the state of a database at a specific time, offload reporting, or maintain historical data. For instance, an organization can revert to a snapshot in the event it runs into problems. Keep in mind that this feature is available only with SQL Server 2005 Enterprise Edition.

Caution

It is important to note here that database snapshots are a convenient feature, but they are not a replacement for maintaining a backup and restore plan. A backup and recovery plan is still necessary because changes made after the snapshot is taken are lost unless an organization also maintains regular and full backups that include the latest transaction log. This ensures that the most recent data is available for recovery. As a result, a snapshot should be viewed only as an extra layer of protection for preserving a database to a specific point in time.

If you want to fully use and manage snapshots, two tools are necessary. You must use TSQL to create and revert to snapshots and SSMS to view snapshots. Both TSQL and SSMS can be used to delete snapshots. The following sections show how to create, view, revert to, and delete database snapshots.

Creating a Database Snapshot with TSQL

In the Query Analyzer, execute the following script to create a database snapshot for the AdventureWorks database:

```
--Creating a database Snapshot with Transact-SQL
USE AdventureWorks
CREATE DATABASE AdventureWorks_SS_09_05_1974 ON
( NAME = AdventureWorks_Data, FILENAME =
'X:\SQLSnapShots\AdventureWorks_DS_09_05_1974.ss' )
AS SNAPSHOT OF AdventureWorks;
GO
```

Viewing a Database Snapshot with SSMS

After creating the database snapshot, you can view it using SSMS. Follow the steps in the preceding section to view the AdventureWorks database snapshot and then follow these steps to view the snapshot:

1. In Object Explorer, first connect to the Database Engine, expand the desired server, expand the Database folder, and then expand the Database Snapshots folder.
2. Select the desired database snapshot to view (for this example, AdventureWorks_SS_09_05_1974), as shown in Figure 17.17.

Reverting to a Database Snapshot with TSQL

In Query Analyzer, execute the following script to revert the AdventureWorks database with the database snapshot created in the preceding steps:

```
USE Master
RESTORE DATABASE AdventureWorks FROM DATABASE_SNAPSHOT
➥= 'AdventureWorks_SS_09_05_1974'
GO
```

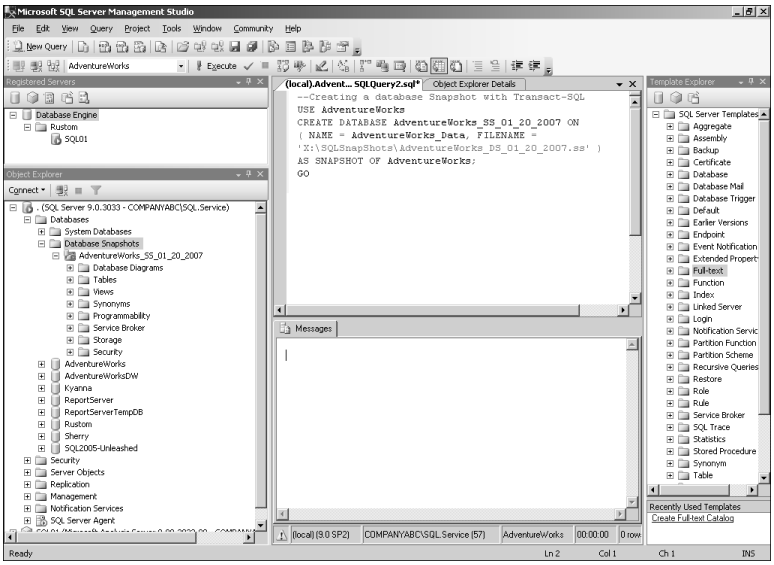

FIGURE 17.17
Viewing database snapshots with SSMS.

Dropping a Database Snapshot

You can drop a database snapshot by either right-clicking on the snapshot in SSMS and choosing Delete or by issuing a drop statement that identifies the name of the snapshot to be dropped.

The following script drops the AdventureWorks snapshot created in the preceding steps:

```
DROP DATABASE AdventureWorks_SS_09_05_1974
```

For more information on database snapshots, see the Database Snapshot chapter in *Microsoft SQL Server 2005 Unleashed* (Sams Publishing, ISBN: 0-672-32824-0).

Summary

Although you, as database administrator, are charged with maintaining the SQL Server database, you can easily get caught up in daily administration and firefighting because your workload is often laborious. It is nonetheless imperative for you to develop and implement a strategic SQL Server backup

and recovery plan. Currently, the backup maintenance task is likely the easiest tool available to implement the backup strategy for all user and system relational databases. In the event Analysis Services, Reporting Services, or Internet Information Services is being used, it is important for each component to find its way into your organization's backup and restore plan. Finally, your organization should make it a habit to test and document your backup and recovery plan prior to implementation.

Best Practices

The following are best practices for backing up and restoring SQL Server 2005:

- Define and document a SLA relevant to the SQL Server 2005 environment.
- Test the backup and recovery plan on a periodic basis and also before production to validate it is operational and the SLA can be met.
- Select the appropriate recovery model for all systems and user databases. Use the Full recovery model for mission-critical databases that need to be restored to the point of failure in the event of a disaster.
- Isolate database and transaction log files on separate spindles for recovery purposes.
- Save the backups locally on a redundant disk drive separate from the online databases and back up to tape on a regular basis.
- If database or transaction log backups are stored locally on disk, do *not* store them on the same volumes as the database and transaction log files. If a drive or volume failure occurs, both the files and backups could be lost.
- For retention and recovery purposes, commit to tape the backups stored to disk.
- Commit to doing frequent backups if the system is an online transaction processing (OLTP) environment. An OLTP database is also known as the databases residing in the SQL Server database engine.
- Try to schedule backups when SQL Server is not in the process of being heavily updated.
- Use maintenance plans to streamline, automate, and schedule backups for all system and user databases. Don't forget to leverage the new features included with Service Pack 2.

- For large databases, consider introducing additional files or filegroups and include a combination of full, differential, filegroup, and transaction log backups to reduce backup and restore times.
- Speed up the backup process by selecting multiple backup devices.
- Do *not* forget to include all the SQL Server components that are installed when creating a backup and recovery plan.
- When restoring the database to the point in failure, remember to first back up the tail-log and then conduct the restore.
- When backing up databases in Analysis Services, do *not* forget to also back up the associated relational databases depending on the OLAP storage model used.