

Module 7: Restoring Databases

Contents

Overview	1
SQL Server Recovery Process	2
Preparing to Restore a Database	5
Restoring Backups	9
Restoring Databases from Different Backup Types	14
Restoring Damaged System Databases	24
Recommended Practices	26
Lab A: Restoring Databases	27
Review	38



Information in this document is subject to change without notice. The names of companies, products, people, characters, and/or data mentioned herein are fictitious and are in no way intended to represent any real individual, company, product, or event, unless otherwise noted. Complying with all applicable copyright laws is the responsibility of the user. No part of this document may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Microsoft Corporation. If, however, your only means of access is electronic, permission to print one copy is hereby granted.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2000 Microsoft Corporation. All rights reserved.

Microsoft, Active Directory, ActiveX, BackOffice, FrontPage, Jscript, Outlook, PowerPoint, Visual Basic, Visual Studio, Windows, Windows Media, and Windows NT are either registered trademarks or trademarks of Microsoft Corporation in the U.S.A. and/or other countries.

Other product and company names mentioned herein may be the trademarks of their respective owners.

Development Lead: Xandria Eykel

Technical Lead: Rick Byham

Instructional Designers: Cheryl Hoople, Lin Joyner (Content Master Ltd), Marilyn McGill (Independent Contractor), Gordon Ritchie (Content Master Ltd.),

Subject Matter Experts: Karl Dehmer, Mike Galos, Graeme Malcolm (Content Master), Mary Neville (Content Master Ltd), and Carl Rabeler (Shadow Mountain Computers),

Classroom Automation: Lorrin Smith-Bates

Graphic Artist: Kimberly Jackson (Independent Contractor)

Editing Manager: Lynette Skinner

Editor: Wendy Cleary

Copy Editor: Bill Jones (S&T Consulting)

Production Manager: Miracle Davis

Production Coordinator: Jenny Boe

Production Support: Ed Casper (S&T Consulting), Theano Petersen (S&T Consulting)

Test Manager: Sid Benavente

Courseware Testing: Testing Testing 123

Creative Director, Media/Sim Services: David Mahlmann

Web Development Lead: Lisa Pease

CD Build Specialist: Julie Challenger

Online Support: David Myka (S&T Consulting)

Localization Manager: Rick Terek

Operations Coordinator: John Williams

Manufacturing Support: Laura King; Kathy Hershey

Lead Product Manager, Release Management: Bo Galford

Lead Product Manager, Database Management: Margo Crandall

Group Manager, Courseware Infrastructure: David Bramble

Group Product Manager, Content Development: Dean Murray

General Manager: Robert Stewart

Instructor Notes

Presentation:
60 Minutes

Lab:
45 Minutes

This module provides students with the knowledge and skills that they need to restore databases, transaction logs, files or filegroups, and damaged system databases. Students will learn about the Microsoft® SQL Server™ 2000 recovery process and how they can use the RESTORE statement to get information and perform restore operations. Students will be able to determine how to perform restore operations based on a particular backup method.

In the lab, students will have an opportunity to restore databases and transaction logs, and recover data from a media failure.

After completing this module, students will be able to:

- Describe the SQL Server recovery process.
- Verify backups and perform specific tasks that enable the restore process.
- Use the RESTORE statement to get information about a backup file before you restore a database, file, or transaction log.
- Restore backups from different backup types and use the appropriate options.
- Restore damaged system databases.

Materials and Preparation

This section provides the materials and preparation tasks that you need to teach this module.

Required Materials

To teach this module, you need the following materials:

- Microsoft PowerPoint® file 2072A_07.ppt.
- The C:\Moc\2072A\Demo\D07_Ex.sql example file, which contains all of the example scripts from the module, unless otherwise noted in the module.

Preparation Tasks

To prepare for this module, you should:

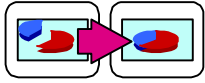
- Read all of the materials for this module.
- Complete the lab.
- Complete all demonstrations.
- Practice the presentation, including the animated slide.
- Review any relevant white papers located on the Trainer Materials compact disc.

Prerequisites

You must teach this module after you have already taught module 6, “Backing Up Databases”, in course 2072A, *Administering a Microsoft SQL Server 2000 Database*. Module 6 discusses the fundamentals of backing up SQL Server databases and the backup strategy most appropriate for particular business environments.

Other Activities

This section provides procedures for implementing interactive activities to present or review information, such as games or role playing exercises.



Displaying the Animated PowerPoint Slides

All animated slides are identified with an icon of links on the lower left corner of the slide.

► To display the Specifying a Point in Time slide

1. Display the topic slide; the series of **Northwind** database backups and the restore process appear.
2. Advance to the next animation where the restore example appears.

Discuss the example and the steps to restore the database to a specific point in time. Point out that the series of transaction logs between the last full database backup and the differential backup do not have to be restored.

Module Strategy

Use the following strategy to present this module:

- SQL Server Recovery Process

Explain the SQL Server recovery process. Point out that this process automatically occurs and can be initiated manually when students restore backups. Then, describe the activities that occur when the recovery process is initiated.

- Preparing to Restore a Database

Discuss the activities that must occur prior to restoring a database, such as obtaining information about the backup and performing specific tasks to prepare to restore backups.

- Restoring Backups

Introduce the RESTORE statement and emphasize the use of the RECOVERY and NORECOVERY options. Then, focus on the options that are commonly used for restore operations.

- Restoring Databases from Different Backup Types

Describe the specific uses and considerations for restoring from a full database, differential, transaction log, and file or filegroup backup. Highlight the feature of restoring transaction logs to a specific point in time.

- Restoring Damaged System Databases

Inform students of the possibility of having to restore or rebuild damaged system databases and describe the tasks that they must perform in order to do so. Emphasize that user databases can either be attached or restored depending on whether the **master** database was rebuilt.

Customization Information

This section identifies the lab setup configuration changes that occur on student computers during the lab. This information is provided to assist you in replicating or customizing Microsoft Official Curriculum (MOC) courseware.

Important The lab in this module is dependent on the classroom configuration that is specified in the Customization Information section at the end of the *Classroom Setup Guide* for course 2072A, *Administering a Microsoft SQL Server 2000 Database*.

Lab Setup

The lab in this module requires that SQL Server 2000 Enterprise Edition has been installed on student computers. To prepare student computers to meet this requirement, perform exercise 1 in lab A, “Installing SQL Server,” in module 2, “Planning to Install SQL Server” of course 2072A, *Administering a Microsoft SQL Server 2000 Database*.

Lab Results

There are no configuration changes on student computers that affect replication or customization.

Overview

Topic Objective

To provide an overview of the module topics and objectives.

Lead-in

In this module, you will learn about restoring databases.

- **SQL Server Recovery Process**
- **Preparing to Restore a Database**
- **Restoring Backups**
- **Restoring Databases from Different Backup Types**
- **Restoring Damaged System Databases**

*******ILLEGAL FOR NON-TRAINER USE*******

This module provides you with the knowledge and skills to restore databases, transaction logs, files or filegroups, and damaged system databases. You will learn about the Microsoft® SQL Server™ 2000 recovery process, how to use the RESTORE statement to get information and perform restore operations, and be able to determine how to perform restore operations based on a particular backup method.

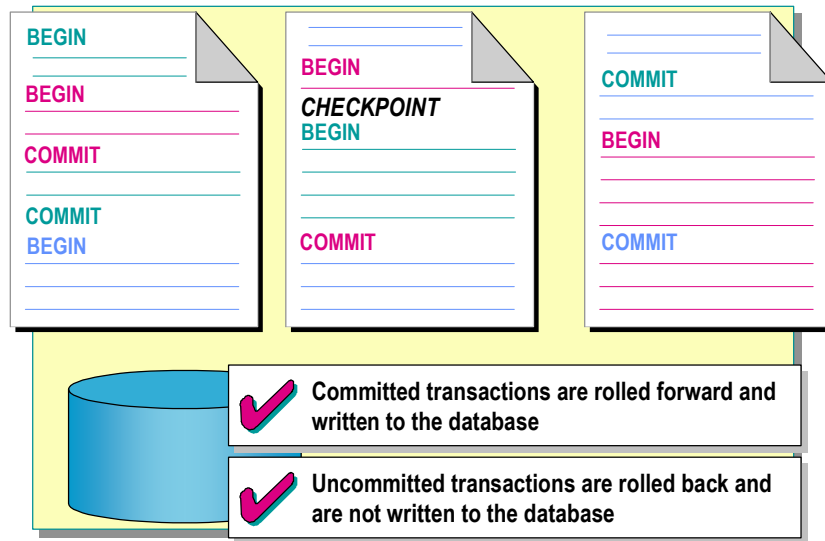
After completing this module, you will be able to:

- Describe the SQL Server 2000 recovery process.
- Verify backups and perform specific tasks that enable the restore process.
- Use the RESTORE statement to get information about a backup file before you restore a database, file, or transaction log.
- Restore backups from different backup types and use the appropriate options.
- Restore damaged system databases.

◆ SQL Server Recovery Process

Topic Objective
To explain the SQL Server automatic recovery process.

Lead-in
The SQL Server recovery process is an internal mechanism that ensures that your database is consistent.



*****ILLEGAL FOR NON-TRAINER USE*****

Key Points
Discuss the status of the transactions in the transaction log at checkpoint.

Then, point out that the recovery process occurs automatically when SQL Server is started and can be initiated manually during restore operations.

The SQL Server recovery process is an internal mechanism that ensures that your database is consistent by examining the transaction log and taking appropriate actions:

- SQL Server examines the transaction log from the last checkpoint to the point at which SQL Server failed or was shut down. A checkpoint is like a bookmark, marking the point at which all data changes are written to the database.
- If the transaction log has committed transactions that have not yet been written to the database, SQL Server rolls these transactions forward, applying the changes to the database.
- If the transaction log contains any uncommitted transactions, SQL Server rolls back these transactions. Uncommitted transactions are not written to the database.

Delivery Tip
Point out that SQL Server performs the checkpoint operation during a normal shutdown.

Occurs Automatically

When your system is restarted after a failure or shutdown, SQL Server begins the automatic recovery process to ensure data consistency. You do not have to start this process manually—it occurs automatically.

Can Be Initiated Manually

You can manually initiate the recovery process when you perform restore operations. The recovery process that you initiate is similar to the automatic recovery process that occurs when SQL Server is restarted.

SQL Server Activities During the Restore Process

Topic Objective

To discuss what activities occur during the restore process.

Lead-in

When you restore databases, SQL Server automatically performs certain actions to ensure that your database is restored quickly.

- **Performs a Safety Check**
 - Database already exists
 - Database files are different
 - Database files are incomplete
- **Recreates the Database and All Associated Files**

*****ILLEGAL FOR NON-TRAINER USE*****

When you restore databases, SQL Server automatically performs certain actions to ensure that your database is restored quickly and with minimal impact on production activities.

Performs a Safety Check

SQL Server performs a safety check when you execute the RESTORE DATABASE statement. This internal mechanism prevents you from accidentally overwriting an existing database with a backup of a different database or with incomplete information.

SQL Server does not restore the database if:

- The database that is named in the RESTORE DATABASE statement already exists on the server, and the database name that is recorded in the backup file is different than the database that is named in the RESTORE DATABASE statement.
- The set of database files on the server is different from the set of database files that are contained in the backup set.
- All of the files that are needed to restore a database or filegroup are not supplied. SQL Server generates an error message, specifying which files must be restored as a unit (in one restore operation).

For example, if you attempt to restore a backup of the **Northwind** database to a database named **Accounting**, and **Accounting** already exists on the server, SQL Server will prevent the restore from occurring. If you intend to restore a backup of **Northwind** and overwrite the data in **Accounting**, you must override the safety check.

Recreates the Database and All Associated Files

When you restore a database from a full database backup, SQL Server recreates the original database files and places them in the locations that were recorded when the backup was made. All database objects are recreated automatically. You do not need to rebuild the database schema before you restore the database.

◆ Preparing to Restore a Database

Topic Objective

To provide an overview of preparing to restore a database.

Lead-in

We will now discuss things that you should do before restoring a database.

- **Verifying Backups**
- **Performing Tasks Before Restoring Backups**

*******ILLEGAL FOR NON-TRAINER USE*******

You should verify the backups to confirm that you are restoring the intended data and objects and that the backup contains valid information.

Before you restore a backup, you must perform specific tasks that enable you to begin the restore process.

Verifying Backups

Topic Objective

To discuss the RESTORE statements that display information about the backup files.

Lead-in

Before you restore a backup file, you must ensure that it is valid.

■ RESTORE HEADERONLY Statement

- Returns header information of a backup file or backup set

■ RESTORE FILELISTONLY Statement

- Returns information about the original database or transaction log files

■ RESTORE LABELONLY Statement

- Returns information about the backup media

■ RESTORE VERIFYONLY Statement

- Verifies that individual files are complete and readable

*****ILLEGAL FOR NON-TRAINER USE*****

Delivery Tip

Use SQL Server Enterprise Manager to demonstrate how to obtain information about the backups *before* you restore them.

Before you restore a backup file, you must ensure that it is valid and that the file contains the expected backups. You can use SQL Server Enterprise Manager to view the property sheet for each backup device. For more detailed information about the backups, you can execute the following Transact-SQL statements.

RESTORE HEADERONLY Statement

Use this statement to obtain the header information of a particular backup file or backup set. If more than one backup resides on a backup file, SQL Server returns header information for all backups that are contained in that file.

When you execute the RESTORE HEADERONLY statement, you receive the following information:

- The backup file or backup set name and description
- The type of backup media that was used, such as a tape or hard disk
- The backup method, such as a full database, differential, transaction log, or file backup
- The date and time that the backup was performed
- The size of the backup
- The sequence number of a particular backup within a chain of backup files

RESTORE FILELISTONLY Statement

Use this statement to obtain information about the original database or transaction log files that are contained in a backup file. Executing this statement can help you avoid restoring the wrong backup files.

When you execute the RESTORE FILELISTONLY statement, SQL Server returns the following information:

- The logical names of the database and transaction log files
- The physical names of the database and transaction log files
- The type of file, such as a database or a transaction log file
- The filegroup membership
- The backup set size, in megabytes (MB)
- The maximum allowed file size, in MB

RESTORE LABELONLY Statement

Use this statement to obtain information about the backup media that holds a backup file.

Key Point

Point out that SQL Server does not verify the structure of the data that is contained in the backup.

RESTORE VERIFYONLY Statement

Use this statement to verify that the individual files that make up the backup set are complete and that all backups are readable. SQL Server does not verify the structure of the data that is contained in the backup.

Performing Tasks Before Restoring Backups

Topic Objective

To discuss the tasks that you must perform before you restore a backup.

Lead-in

Before you restore backups, you must perform these tasks.

- **Restrict Access to the Database**

- Limit access to members of the `db_owner`, `dbcreator`, or `sysadmin` role

- **Back Up the Transaction Log**

- Ensures database consistency
- Captures changes between the last transaction log backup and when the database was taken offline

*****ILLEGAL FOR NON-TRAINER USE*****

Before you restore backups, you must restrict access to the database and back up the transaction log.

Restrict Access to the Database

It is important to restrict access to a database before restoring it. Set the **Members of db_owner, dbcreator, or sysadmin** database option to true.

Back Up the Transaction Log

You can ensure database consistency if you back up the transaction log before you perform any restore operations:

- The transaction log backup is used to recover the database as the last step in the restore process.
- If you do not back up the transaction log before you restore backups, you lose data modifications that have occurred between the last transaction log backup and the time when the database was taken offline.

Note If you are using SQL Query Analyzer or executing a script from the command prompt to restore backups, you must use the **master** database.

◆ Restoring Backups

Topic Objective

To introduce the basic options that apply to restoring backups.

Lead-in

You can use SQL Server Enterprise Manager or the RESTORE statement and specify the options that are specific to the type of backup that you are restoring.

- Using the RESTORE Statement
- Initiating the Recovery Process
- Specifying Restore Options

*****ILLEGAL FOR NON-TRAINER USE*****

You can use SQL Server Enterprise Manager or the RESTORE statement and specify the options that are specific to the type of backup that you are restoring. You also determine whether to initiate the recovery process after each restore operation.

Using the RESTORE Statement

Topic Objective

To discuss using the RESTORE statement to restore backups.

Lead-in

You should be familiar with the RESTORE statement and the various options that you can use when you restore SQL Server backups.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
```

■ Restoring Damaged User Databases

- You do not need to drop the damaged database
- SQL Server automatically recreates the database files and objects

*****ILLEGAL FOR NON-TRAINER USE*****

You can use the RESTORE statement or SQL Server Enterprise Manager to perform restore operations.

The restore options allow you to specify details on how to restore backups.

Partial Syntax

Key Point

Briefly point out the RESTORE options in the partial syntax. These options are discussed in detail in the following pages.

```
RESTORE DATABASE {database_name | @database_name_var}
[FROM <backup_device> [, ... n]]
[WITH
    [FILE = file_number]
    [[,] MOVE 'logical_file_name' TO 'operating_system_file_name']
    [[,] REPLACE]
    [[,] {NORECOVERY | RECOVERY | STANDBY = undo_file_name}]
    [[,] RESTART]
```

Where <*backup_device*> is

```
{ {backup_device_name | @backup_device_name_var} |
  {DISK | TAPE | PIPE} = {'temp_backup_device' |
    @temp_backup_device_var}
}
```

Example

This example restores the **Northwind** database from a permanent backup file.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
```

Restoring Damaged User Databases

If a database becomes damaged, use the RESTORE statement or SQL Server Enterprise Manager to restore over the existing database. When you restore over a database:

- You do not need to drop the damaged database.
- SQL Server automatically recreates the database files and objects.

Initiating the Recovery Process

Topic Objective

To introduce the recovery options that you must specify when you restore backups.

Lead-in

When you restore databases, you must specify whether to initiate the recovery process.

■ Specifying the RECOVERY Option

- Use with the last backup to be restored
- Allows access to database

■ Specifying the NORECOVERY Option

- Use with all backup files except for the last backup to be restored
- Prevents access to database

*****ILLEGAL FOR NON-TRAINER USE*****

Key Point

Emphasize that the recovery process that the recovery options initiate is similar to the automatic recovery process that occurs when SQL Server is restarted.

You always should specify the RECOVERY or NORECOVERY option to prevent administrative errors during the restore process and to make the statement easier to read. The RECOVERY option is the SQL Server default.

Specifying the RECOVERY Option

Use this option with the last transaction log to be restored or with a full database restore to return the database to a consistent state:

- SQL Server rolls back any uncommitted transactions in the transaction log and rolls forward any committed transactions.
- The database is available for use after the recovery process is complete.

Note Do not use this option if you have additional transaction logs or differential backups that must be restored.

Specifying the NORECOVERY Option

Use this option when you have multiple backups to restore. Consider the following facts when you use the NORECOVERY option:

- Specify the NORECOVERY option for all backups *except* for the last backup to be restored.
- SQL Server neither rolls back any uncommitted transactions in the transaction log nor rolls forward any committed transactions.
- The database is unavailable for use until the database is recovered.

Specifying Restore Options

Topic Objective
To introduce the RESTORE statement and options.

Lead-in
When you use the RESTORE statement, use options to specify how the restore operation is done.

RESTORE option	Description
FILE	Restores a specific backup
	You must specify a file number
RESTART	Continues an interrupted recovery operation
MOVE...TO	Specifies where to restore the backup files
	Use to restore to different disk, server, or standby SQL Server
REPLACE	Replaces an existing database
	SQL Server does not perform a safety check

*****ILLEGAL FOR NON-TRAINER USE*****

When you use the RESTORE statement, use the following options to specify how the restore operation is done.

Using the FILE Option

Use this option to select specific backups from a backup file that contains multiple backups. You must specify a file number that corresponds to the order in which the backup exists within the backup file.

Using the RESTART Option

Use this option to continue an interrupted recovery operation. The recovery will continue from the point at which the previous attempt was interrupted.

Using the MOVE...TO Option

Use this option to specify where to restore the backup files if you are restoring files to a different location, such as a different disk, server, or a standby SQL Server.

Note You can also use the **sp_attach_db** or **sp_attach_single_file_db** system stored procedure to move a database from one server to another by copying the database files and then attaching them to the **master** database.

Using the REPLACE Option

Use the REPLACE option *only* if you want to replace an existing database with data from a backup of a different database. If you use the REPLACE option, SQL Server does not perform a safety check.

By default, SQL Server performs a safety check that ensures that an existing database is *not* replaced if:

- The database already exists on the target server and the database name is different from the name that is recorded in the backup set.
- The set of files in the database is different from the files contained in the backup set. SQL Server ignores differences in file size.

◆ Restoring Databases from Different Backup Types

Topic Objective

To identify the different ways to restore a database based on the type of backup that was performed.

Lead-in

To restore a database, you must know the type of backup method that was used to perform the backup and whether the backup exists.

- Restoring from a Full Database Backup
- Restoring from a Differential Backup
- Restoring a Transaction Log Backup
- Restoring from a File or Filegroup Backup

*****ILLEGAL FOR NON-TRAINER USE*****

To restore a database, you must know the type of backup method that was used to perform the backup and whether the backup exists. Confirm that your backup files contain the backups that you want to restore. Make sure that the backups are valid and that you have all files or tapes that contain the backup set.

Restoring from a Full Database Backup

Topic Objective

To discuss restoring from a full database backup.

Lead-in

When you restore a database from a full database backup, SQL Server recreates the database and all of its associated files and then places them in their original locations.

■ When to Use

- Physical disk is damaged
- Entire database is damaged, corrupted, or deleted
- To maintain an identical copy of database on another SQL Server

■ Specifying Recovery Options

- Initiate the recovery process with the RECOVERY option
- Postpone the recovery process with the NORECOVERY option

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
WITH FILE = 2, RECOVERY
```

*****ILLEGAL FOR NON-TRAINER USE*****

When you restore a database from a full database backup, SQL Server recreates the database and all of its associated files and then places them in their original locations. All database objects are recreated automatically. You do not need to rebuild the database schema before you restore the database.

When to Use

You typically will restore from a full database backup when:

- The physical disk of the database is damaged.
- The entire database is damaged, corrupted, or deleted.
- An identical copy of the database is being restored to a different SQL Server.

Specifying Recovery Options

The RECOVERY option initiates the recovery process so that your database is returned to a consistent state:

- If you implement a full database backup strategy and do not have any transaction log or differential backups, specify the RECOVERY option.
- If any transaction log or differential backups exist, specify the NORECOVERY option to postpone the recovery process until the last backup is restored.

Example

This example assumes that a full backup exists on the permanent **NwindBac** backup file and that two backups are appended to that file. The **Northwind** database is completely replaced by the second backup on the **NwindBac** backup device. Finally, the recovery process returns the database to a consistent state (rolls forward committed changes and rolls back uncommitted activities).

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
WITH FILE = 2, RECOVERY
```

Restoring from a Differential Backup

Topic Objective

To discuss restoring from a differential backup.

Lead-in

When you restore from a differential backup, SQL Server restores only the parts of the database that have changed since the last full database backup.

- Restores Only the Parts of the Database That Have Changed Since the Last Full Database Backup
- Returns the Database to the Exact State It Was in When the Differential Backup Was Performed
- Takes Less Time Than Applying a Series of Transaction Logs

Syntax is the same as when you restore a full database

Specify the backup file that contains the differential backup

```
USE master
RESTORE DATABASE Northwind
FROM NwindBacDiff
WITH NORECOVERY
```

*****ILLEGAL FOR NON-TRAINER USE*****

When you restore a database from a differential database backup, SQL Server:

- Restores only the parts of the database that have changed since the last full database backup.
- Returns the database to the exact condition it was in when the differential backup was performed.
- Often takes less time to restore differential backups than it does to apply a series of transaction logs that represent the same database activity.

Considerations for Restoring Differential Backups

When you restore from a differential backup, consider the following facts and guidelines:

- Restore the full database backup *before* you restore a differential backup.
- The syntax for restoring a differential backup is the same as for restoring a full database backup. Rather than specifying the full database backup in the FROM clause, specify the backup file that contains the differential backup.
- Specify the NORECOVERY option when there are remaining transaction logs to be restored; otherwise, specify the RECOVERY option.

Example

The following example restores a differential backup without recovering the database. The **NwindBacDiff** file contains a differential backup. SQL Server allows you to restore transaction logs before you bring the **Northwind** database to a consistent state by specifying the NORECOVERY option.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBacDiff
WITH NORECOVERY
```

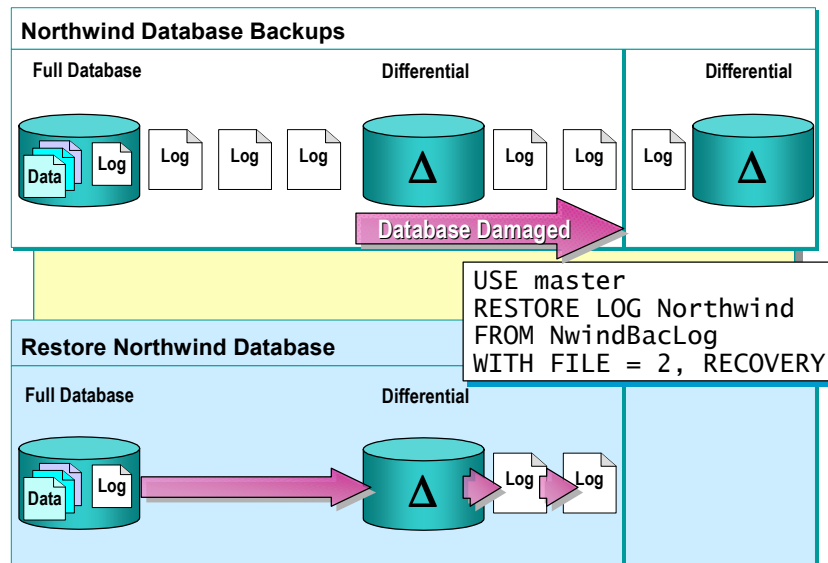
◆ Restoring a Transaction Log Backup

Topic Objective

To illustrate the purpose of restoring transaction log backups.

Lead-in

When you restore from a transaction log backup, SQL Server restores changes to the database that are recorded in the transaction log.



*****ILLEGAL FOR NON-TRAINER USE*****

Key Points

Point out that the syntax for restoring transaction logs is similar to that for restoring databases, with one exception: the STOPAT option specifies a point in time.

The next slide discusses the STOPAT option in detail.

When you restore from a transaction log backup, SQL Server restores changes to the database that are recorded in the transaction log.

You typically will restore transaction logs in order to apply changes that are made to the database since the last full database or differential backup. More importantly, you can restore transaction logs in order to recover a database up to a specific point in time.

Considerations for Restoring Transaction Logs

Although restoring a differential backup may speed up the restore process, in order to ensure data consistency you may have to restore additional transaction log backups that were created after a differential backup.

Before you restore any transaction logs, you first must restore the full database backup. When you have multiple transaction logs to apply, specify the NORECOVERY option for all transaction logs except the last one. SQL Server suspends the recovery process until the last transaction log is restored.

Partial Syntax

```
RESTORE LOG {database_name | @database_name_var}
[FROM <backup_device> [, ...n]]
[WITH
  [{NORECOVERY | RECOVERY | STANDBY = undo_file_name}]
  [[,] STOPAT = {date_time | @date_time_var}]
  [[,] STOPBEFOREMARK = mark_name [AFTER date_time]]
  [[,] STOPATMARK = mark_name [AFTER date_time]]
```


If you are planning a high-risk operation, you may want to add a log mark so that you can restore the database to the point before the operation began. To restore to a specified log mark:

Delivery Tip

Point out that it is possible to restore to a specified log mark.

- Use the `WITH STOPATMARK='mark_name'` clause to roll forward to the mark and include the transaction that contains the mark.
- Use the `WITH STOPBEFOREMARK='mark_name'` clause to roll forward to the mark and exclude the transaction that contains the mark.

Example

This example assumes that a full database exists on one backup file and that two transaction log backups exist on another backup file. Three separate restore operations are performed to ensure database consistency.

Delivery Tip

Go through the steps that are provided in this example to discuss the order in which the backups must be restored.

1. The first step restores from a full database backup without recovering the database.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
WITH NORECOVERY
```

2. The second step restores the first transaction log without recovering the database. The progress of the restore process is displayed.

```
USE master
RESTORE LOG Northwind
FROM NwindBacLog
WITH FILE = 1,
    STATS,
    NORECOVERY
```

3. The third step restores the second transaction log, rolls forward any committed transactions, and rolls back any uncommitted transactions. The `RECOVERY` option returns the **Northwind** database to a consistent state.

```
USE master
RESTORE LOG Northwind
FROM NwindBacLog
WITH FILE = 2,
    RECOVERY
```

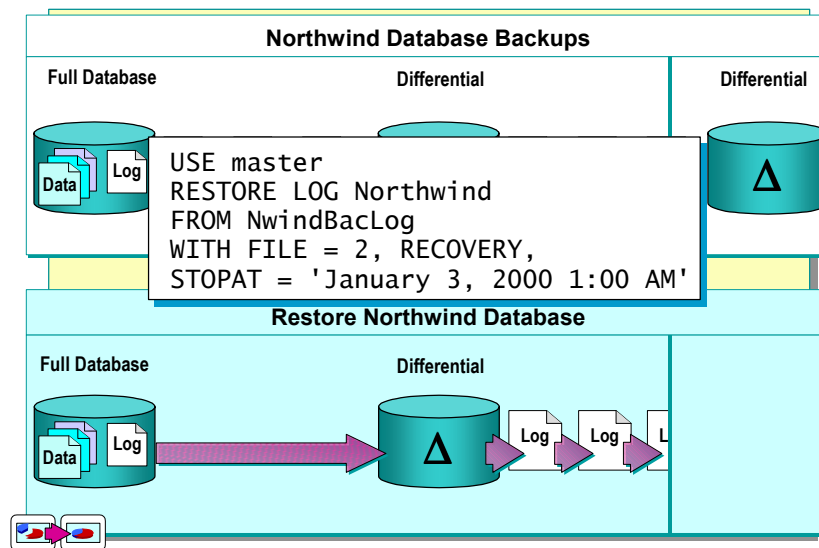
Specifying a Point in Time

Topic Objective

To discuss restoring transaction logs to a specific point in time.

Lead-in

When you restore transaction logs, you can restore to a specific point in time by using the STOPAT option.



*****ILLEGAL FOR NON-TRAINER USE*****

Delivery Tip

This slide is animated. Refer to the Instructor Notes if you require help in navigating through this slide.

When you restore transaction logs, you can restore to a specific point in time by using the STOPAT option:

- Use the STOPAT option to recover a database in the state that it was in at the exact moment before data corruption or some other event occurred.

For example, if you know that a malicious update to a database occurred at 11:00 A.M., you can restore the changes in the transaction log through 10:59 A.M. and not apply any changes that occurred after that point.

- You must specify the date and time to stop loading a backup onto the database.

SQL Server restores all of the transaction log records that were written to the database *before* a specific point in time.

Note You cannot use the STOPAT option with the STANDBY or NORECOVERY option.

Example

This example assumes that a full database and two transaction log backups exist on one backup file. However, only changes that occurred before 1:00 A.M. on January 3, 2000 are restored. Three separate restore operations are performed to ensure database consistency.

Delivery Tip

Go through the steps that are provided in this example to discuss the order in which the backups must be restored.

1. The first step restores a database from a full database backup without recovering the database.

```
USE master
RESTORE DATABASE Northwind
FROM NwindBac
WITH NORECOVERY
```

2. The second step restores a transaction log without recovering the database.

```
USE master
RESTORE LOG Northwind
FROM NwindBacLog
WITH FILE = 1,
     NORECOVERY
```

3. The third step restores a second transaction log, applies changes that occurred before 1:00 A.M. on January 3, 2000, and recovers the database.

```
USE master
RESTORE LOG Northwind
FROM NwindBacLog
WITH FILE = 2,
     RECOVERY,
     STOPAT = 'January 3, 2000 1:00 AM'
```

Restoring from a File or Filegroup Backup

Topic Objective

To discuss how to restore from a file or filegroup backup.

Lead-in

When you restore from a file or filegroup, you must:

- **Apply All Transaction Logs Since the File Backup**
- **Restore Filegroup Backups Containing Indexes and Tables as One Unit**

```
USE master
RESTORE DATABASE Northwind
FILE = Nwind2
FROM Nwind2Bac WITH NORECOVERY
```

*****ILLEGAL FOR NON-TRAINER USE*****

Key Point

Point out that restoring from a file or filegroup is beneficial for very large databases and when only specific files are deleted or damaged.

SQL Server allows you to restore a database file from a full database backup or an individual file backup. If you have file or filegroups backups, you can restore them to:

- Reduce the time that is required to restore a very large database (VLDB).
- Recover data when a particular file was accidentally deleted or damaged.

Considerations for Restoring File or Filegroup Backups

When you restore from a file or filegroup, you must:

- Apply all transaction logs that were created since the file backup was created in order to bring the restored file or filegroup into a state that is consistent with the rest of the database.
SQL Server only applies transactions from log backups that affect the restored file.
- Restore the filegroup backups as a unit if a table and its associated indexes exist on two different filegroups.

Partial Syntax

```
RESTORE DATABASE {database_name | @database_name_var}
<file_or_filegroup> [, ...m]
[FROM <backup_device> [, ...n]]
```

Where <*file_or_filegroup*> is

```
{FILE = logical_file_name | FILEGROUP = logical_filegroup_name}
```

Example

This example assumes that a database exists on three files: Nwind1, Nwind2, and Nwind3. The Nwind2 database file contains a single table and its related indexes. The Nwind2 database file was backed up onto the Nwind2bac backup file. One transaction log backup was performed since Nwind2 was last backed up. Nwind2 must be restored because the physical media is damaged. The example consists of two steps to ensure database consistency.

1. The first step restores the backup of the Nwind2 database file without rolling forward any committed transactions or rolling back any uncommitted transactions.

```
USE master
RESTORE DATABASE Northwind
    FILE = Nwind2
FROM Nwind2Bac
WITH NORECOVERY
```

2. The second step restores the transaction log backup, rolls forward any committed transactions, and rolls back any uncommitted transactions.

```
USE master
RESTORE LOG Northwind
FROM NwindBacLog
WITH RECOVERY
```

Restoring Damaged System Databases

Topic Objective

To explain the process of restoring damaged system databases.

Lead-in

If the media that contain the system databases are damaged, you may have to rebuild the system databases.

- **Restoring System Databases from a Backup**
- **Rebuilding System Databases**
- **Attaching or Restoring User Databases**
 - Restore from a backup
 - Attach with **sp_attach_db** or **sp_attach_single_file_db** system stored procedure

*****ILLEGAL FOR NON-TRAINER USE*****

If the media that contain the system databases are damaged, you may have to rebuild the system databases.

Restoring System Databases from a Backup

If the SQL Server service can be started, you can use the RESTORE DATABASE statement or SQL Server Enterprise Manager to restore the system databases from a valid backup.

Rebuilding System Databases

If the **master** database is damaged and you cannot start SQL Server, perform the following steps:

1. Rebuild the system databases with the **Rebuildm.exe** command-prompt utility that is stored in 80/Tools/Binn.
2. Restart the SQL Server service.
3. Restore backups of the system databases.

Note Rebuilding the system databases overwrites the existing **master**, **model**, and **msdb** databases.

Restoring System Databases

After the system databases are rebuilt and SQL Server is started, you should perform the following steps:

1. Restore the **master** database from a backup, if one exists.

If a valid backup of **master** does not exist, you must recreate the information that is stored in **master** manually by using one of the following:

- SQL Server Enterprise Manager
- The original scripts that you used to create the **master** database, which includes references to:
 - User databases.
 - Database files.
 - Backup devices.
 - SQL Server login accounts and server-wide security roles.

2. Restore the **msdb** database from a backup, if one exists.

You must restore the **msdb** database when you rebuild the **master** database. When you run the **rebuildm** utility, the **msdb** database is dropped and then recreated. Therefore, all scheduling information is lost.

3. Restore the **model** database from a backup, if one exists.

Attaching or Restoring User Databases

You either attach or restore user databases depending on whether the **master** database was rebuilt:

- If the **master** database was restored from a valid backup, it will contain references to each user database. No further action is needed.
- If the **master** database was rebuilt and a valid backup was not applied, you must perform one of the following:
 - Restore the user databases from a backup
 - Attach the existing user database files to the new master database

If the user database files are not damaged, attach them to the new **master** database by using the **sp_attach_db** or **sp_attach_single_file_db** system stored procedure.

Attaching existing database files informs the **master** database about a user database. You do not need a backup of a database to attach it to the **master** database.

Note Attaching a user database is more efficient than restoring from a backup.

Example

This example attaches the **Northwind** database to the **master** database.

```
USE master
EXEC sp_attach_single_file_db @dbname = 'Northwind',
@physname = 'Mssql1\Data\Northwind.mdf'
```






Recommended Practices

Topic Objective

To list recommended practices to consider when you restore databases, transaction logs, or files.

Lead-in

The following are recommended practices to consider when you restore databases, transaction logs, or files.

-  **Obtain Information About Backups Before You Restore**
-  **Specify NORECOVERY on All Except the Last Backup**
-  **Use RECOVERY on the Last Backup to Return the Database to a Consistent State**
-  **Add a Log Mark Before Performing a High-Risk Operation**
-  **Test Backup Files Periodically By Using RESTORE VERIFYONLY**

*****ILLEGAL FOR NON-TRAINER USE*****

Consider the following recommended practices when you restore databases, transaction logs, or files:

- You should obtain information about the backups that you plan to restore. You must ensure that the files are valid and contain all of the backups that are required to restore the database to a consistent state.
- You should use the NORECOVERY option if you have additional backups that must be restored.
- You always should use the RECOVERY option on the last backup to return the database to a consistent state.
- If you are planning a high-risk operation, you may want to add a log mark so that the database can be restored to the point before the operation began.
- Test your backup files periodically by using the RESTORE VERIFYONLY statement.

Additional information on the following topics is available in SQL Server Books Online.

Topic	Search on
Using the RESTORE statement to obtain backup information	“restore headeronly” “restore filelistonly” “restore labelonly” “restore verifyonly”
Restoring from different types of backups	“how to restore”
Moving databases	sp_attach_db

Lab A: Restoring Databases

Topic Objective

To introduce the lab.

Lead-in

In this lab, you will restore a database, transaction logs, and files and recover from a media failure.



*****ILLEGAL FOR NON-TRAINER USE*****

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Restore a database from a full database backup.
- Retrieve information about backup sets.
- Restore differential and transaction log backups.

Prerequisites

Before working on this lab, you must have script files for this lab, which are located in C:\Moc\2072A\Labfiles\L07.

For More Information

If you require help in executing files, search SQL Query Analyzer Help for “Execute a query”.

Other resources that you can use include:

- The **Northwind** database schema.
- SQL Server Books Online.

Scenario

The organization of the classroom is meant to simulate that of a worldwide trading firm named Northwind Traders. Its fictitious domain name is nwtraders.msft. The primary DNS server for nwtraders.msft is the instructor computer, which has an Internet Protocol (IP) address of 192.168.x.200 (where *x* is the assigned classroom number). The name of the instructor computer is London.

The following table provides the user name, computer name, and IP address for each student computer in the fictitious nwtraders.msft domain. Find the user name for your computer, and make a note of it.

User name	Computer name	IP address
SQLAdmin1	Vancouver	192.168.x.1
SQLAdmin2	Denver	192.168.x.2
SQLAdmin3	Perth	192.168.x.3
SQLAdmin4	Brisbane	192.168.x.4
SQLAdmin5	Lisbon	192.168.x.5
SQLAdmin6	Bonn	192.168.x.6
SQLAdmin7	Lima	192.168.x.7
SQLAdmin8	Santiago	192.168.x.8
SQLAdmin9	Bangalore	192.168.x.9
SQLAdmin10	Singapore	192.168.x.10
SQLAdmin11	Casablanca	192.168.x.11
SQLAdmin12	Tunis	192.168.x.12
SQLAdmin13	Acapulco	192.168.x.13
SQLAdmin14	Miami	192.168.x.14
SQLAdmin15	Auckland	192.168.x.15
SQLAdmin16	Suva	192.168.x.16
SQLAdmin17	Stockholm	192.168.x.17
SQLAdmin18	Moscow	192.168.x.18
SQLAdmin19	Caracas	192.168.x.19
SQLAdmin20	Montevideo	192.168.x.20
SQLAdmin21	Manila	192.168.x.21
SQLAdmin22	Tokyo	192.168.x.22
SQLAdmin23	Khartoum	192.168.x.23
SQLAdmin24	Nairobi	192.168.x.24

Estimated time to complete this lab: 45 minutes

Exercise 1

Restoring from a Full Database Backup

In this exercise, you will review and execute multiple scripts that modify, back up, and intentionally corrupt the **NWCOPY** database. You will use SQL Server Enterprise Manager to restore the **NWCOPY** database from a full database backup and confirm that data was recovered.

► To create the NWCOPY database

1. Log on to the **NWTraders** classroom domain by using the information in the following table.

Option	Value
User name	SQLAdminx (where <i>x</i> corresponds to your computer name as designated in the nwtraders.msft classroom domain)
Password	password

2. Copy the Labfiles\L07\NWC1.bak file to C:\Backup on your local hard disk.
3. Open SQL Query Analyzer and, if requested, log in to the (local) server with Windows Authentication.

You have permission to log in to and administer SQL Server because you are logged as **SQLAdminx**, which is a member of the Windows 2000 local group, Administrators. All members of this group are automatically mapped to the SQL Server **sysadmin** role.

4. Execute Labfiles\L07\SetupNWC.sql.

This script restores the **NWCOPY** database that is used in this lab.

► To modify the NWCOPY database

In this procedure, you will execute a script that adds a row to the **Products** table. You then will write and execute a query that returns the new row.

1. Open a query window, open Labfiles\L07\AddProd.sql, review its contents, and then execute it.

This script adds the new product Maple Flavor Pancake Mix to the **Products** table.

2. Review the results to confirm that the new row was added.

► To back up the NWCOPY database

In this procedure, you will execute a script that backs up the **NWCOPY** database to a single backup file.

- Open Labfiles\L07\MakeBack.sql, review its contents, and then execute it.

This script backs up the **NWCOPY** database to a single backup file. This backup file has a logical name of NWC2 and a physical name of C:\Backup\NWC2.bak.

► **To simulate accidental data modification**

In this procedure, you will execute a script that damages the database by updating all rows in the **Products** table. You then will write and execute a query to confirm that the product name Maple Flavor Pancake Mix was removed from the **Products** table.

1. Open a query window, open Labfiles\L07\DataLoss.sql, review its contents, and then execute it.

This script damages the database by updating all rows in the **Products** table.

2. Review the result to confirm that the product name Maple Flavor Pancake Mix was removed from the **Products** table.
3. Close the query window.

Important You must close the query window and break the connection to the **NWCOPY** database in order to perform the next procedure. The restore operation requires that no users be connected to the database.

► **To restore the NWCOPY database from a full database backup**

In this procedure, you will use SQL Server Enterprise Manager to restrict access to the **NWCOPY** database, restore from a full database backup, and then allow access to the database after the restore process is complete.

1. Open SQL Server Enterprise Manager.
2. In the console tree, expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
3. Expand your server, expand **Databases**, right-click **NWCOPY**, and then click **Properties**.
4. On the **Option** tab, select **Restrict Access**, select the **Members of db_owner, dbcreator, or sysadmin** option to restrict access to the database during the restore process, and then click **OK** to continue.
5. In the console tree, right click **NWCOPY**, point to **All Tasks**, and then click **Restore Database**.
6. On the **General** tab, in the **Restore as Database** field, type the name of the database to restore.
7. On the **General** tab, select **Database**.
8. In the **First backup to restore** list, select **<Date Time> - NWCOPY-Full**.
9. In the **Restore** list, select the database backup to restore.
10. On the **Options** tab, select **Leave database operational. No additional transaction logs can be restored**.
11. Click **OK** to close the dialog box and restore the **NWCOPY** database.

► **To confirm that data was recovered**

In this procedure, you will write and execute a query that returns a list of all products in the **Products** table, including the new product Maple Flavor Pancake Mix.

- Open a query window. Write and execute the following query to confirm that all products, including Maple Flavor Pancake Mix, are included in the list:

```
USE NWCOPY
SELECT *
FROM Products
```

Exercise 2

Simulating and Capturing Database Activity

In this exercise, you will review and execute multiple files that add data to the **NWCOPY** database and perform full database, differential, and transaction log backups. You will simulate damaging the device that contains the **NWCOPY** database and review the error message in the Microsoft Windows NT® application log.

► To perform a full database backup of the **NWCOPY** database

In this procedure, you will execute a script that backs up the **NWCOPY** database to the **NWC3** disk backup device.

- Open a query window, open Labfiles\L07\FullBack.sql, review its contents, and then execute it. This script backs up the **NWCOPY** database to the **NWC3** disk backup device.

► To modify the **NWCOPY** database and back up the transaction log

In this procedure, you will execute a script that adds a customer to the **Customers** table and confirms that the customer was added. Then, you will execute another script that backs up the transaction log to the **NWCHANGE** disk backup device.

1. Open a query window, open Labfiles\L07\AddCust1.sql, review its contents, and then execute it.

This script adds the Health Food Store as a customer to the **Customers** table and queries the table to return the new customer.

2. Open a query window, open Labfiles\L07\LogBack1.sql, review its contents, and then execute it.

This script backs up the transaction log of the **NWCOPY** database to the **NWCHANGE** disk backup device.

► To modify the **NWCOPY** database and perform a differential backup

In this procedure, you will execute a script that adds another customer to the **Customers** table and returns that customer to confirm that the customer was added. Then, you will execute another script that performs a differential backup and appends it to the **NWCHANGE** disk backup device.

1. Open a query window, open Labfiles\L07\AddCust2.sql, review its contents, and then execute it.

This script adds the Volcano Coffee Company to the **Customers** table and queries the table to return the new customer.

2. Open a query window, open Labfiles\L07\DiffBack.sql, review its contents, and then execute it.

This script performs a differential backup to capture all changes since the last full database backup. The differential backup is appended to the **NWCHANGE** disk backup device.

► **To modify the NWCOPY database and perform another transaction log backup**

In this procedure, you will execute a script that adds another customer to the **NWCOPY** database and confirms that the customer was added. Then, you will execute another script that performs a second transaction log backup and appends it to the NWCHANGE disk backup device.

1. Open a query window, open Labfiles\L07\AddCust3.sql, review its contents, and then execute it.

This script adds The Wine Cellar as a customer to the **Customers** table and queries the table to return the new customer.

2. Open a query window, open Labfiles\L07\LogBack2.sql, review its contents, and then execute it.

This script backs up the transaction log of the **NWCOPY** database and appends it to the NWCHANGE disk backup device.

► **To simulate damage to the database**

In this procedure, you will simulate damaging the device that stores the **NWCOPY** database.

1. Open SQL Server Service Manager and stop the SQL Server service.
2. Switch to SQL Server Enterprise Manager and exit.
3. Rename the data file for the **NWCOPY** database from Mssql\Data\Nwcopy_data.mdf to **Nwcopy_data.bad**
4. Restart the SQL Server service.
5. Open SQL Server Enterprise Manager.
6. In the console tree, expand your server, and click **Databases**.

SQL Server indicates that the **NWCOPY** database is suspect.

7. Open the Event Viewer and examine the contents of the application log.

Error messages should be present, stating that the NWCopy_Data.mdf file could not be found and that the device could not be opened.

What should you do to restore and recover the **NWCOPY** database?

If possible, back up the transaction log of the damaged database by using the NO TRUNCATE option to capture the latest activity in the log.

Determine the backups that are available and usable.

Restore the full database backup.

Restore the latest differential backup. This will include the changes in the first transaction log backup.

Restore the last set of transaction log backups and recover the database.

Exercise 3

Restoring Full Database, Differential, and Transaction Log Backups

In this exercise, you will use SQL Server Enterprise Manager to examine all backup devices that contain backups of the **NWCOPY** database and will determine whether the selected backup strategy is appropriate. You will go through a series of restore operations and review the contents of the database after each restore process.

► To examine available backups

In this procedure, you will use SQL Server Enterprise Manager to examine the contents and creation date of all **NWCOPY** database backups.

1. Switch to SQL Server Enterprise Manager.
2. Expand your server, expand **Management**, and then click **Backup**.
3. In the details pane, right-click **NWC3**, and then click **Properties**.
4. Click **View Contents** to examine the contents of the NWC3 device. Notice the type, description, and date and time of each backup on the device.

What does the NWC3 device contain?

The NWC3 device contains a full database backup. The date and time reflect the fact that it was created before the backups on the NWCHANGE device.

5. Examine the contents of the NWCHANGE device. Notice the type, description, and date and time of each backup on the device.

What does the NWCHANGE device contain?

The NWCHANGE device contains two transaction log backups, as well as a differential backup. The time stamps show that a transaction log backup was created first, followed by a differential backup and then another transaction log backup.

► **To review the selected restore strategy**

In this procedure, you will review the restore strategy that SQL Server Enterprise Manager suggests automatically and determine whether it is appropriate.

1. In the console tree, right-click **NWCOPY**, point to **All Tasks**, and then click **Restore Databases**.
2. The **Restore Databases** dialog box appears. Verify that the **NWCOPY** database is selected.

Notice that four backups are listed. SQL Server automatically selects the backups that should be restored to return the database to a consistent state. Three out of four backups are selected (full database, differential, and one transaction log).

Do you agree that the selected backups should be restored?

Yes. It is not necessary to restore the first transaction log backup.

Why is the first transaction log backup not selected?

The changes that are recorded in this transaction log backup are reflected in the differential backup.

3. Click **Cancel** to close the **Restore Database** dialog box.

► **To restore full database and differential backups**

In this procedure, you will use SQL Server Enterprise Manager to restore only the full database and differential backups, and then allow read-only access to the database after the restore process is complete.

1. In the console tree, expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
2. Expand your server, expand **Databases**, right-click the database, point to **All Tasks**, and then click **Restore Database**.
3. Under **Restore as Database**, type **NWCOPY**.
4. On the **General** tab, select **Database**.
5. In the **Show backups of database** list, select **NWCOPY**.
6. In the **First backup to restore** list, select the full database backup.
7. In the **Restore** list, select the full database backup and differential database backup files.
8. On the **Options** tab, select **Leave database read-only and able to restore additional transaction logs**.
9. Click **OK** to begin the restore process.
10. Click **OK** when the restore completes.

► **To examine the contents of a database**

In this procedure, you will execute a script that lists all of the customers in the **Customers** table in order to evaluate the restore process.

1. Open a query window, open Labfiles\L07>ListCust.sql, review its contents, and then execute it.

This script determines whether the three new customers that were added to the **Customers** table in Exercise 2 were recovered.

Have all three new customers been recovered?

No, only the rows for Health Food Store and Volcano Coffee Company are in the database. Both of these rows were recorded in the differential backup. The row for The Wine Cellar was recorded in the last transaction log backup, which was not restored.

2. Close the query window.

► **To restore the transaction log backup**

In this procedure, you will use SQL Server Enterprise Manager to restore the transaction log, and then allow access to the database after the restore process is complete.

1. Switch to SQL Server Enterprise Manager.
2. Expand **Databases**, right click the database, point to **All Tasks**, and then click **Restore Database**.
3. Under **Restore as Database**, type **NWCOPY**.
4. On the **General** tab, select **Database**.
5. In the **Show backups of database** list, select **NWCOPY**.
6. In the **First backup to restore** list, select the full database backup.
7. In the **Restore** list, select the latest transaction log (backup set #3 on the NWCHANGE device).
8. On the **Options** tab, select **Leave database operational. No additional transaction logs can be restored**.
9. Click **OK** to begin the restore process.
10. Click **OK** when the restore completes.
11. In the details pane, right click **NWCOPY**, and then click **Properties**.
12. On the **Option** tab, clear **Restrict Access**, and then click **OK** to continue.

► **To examine the contents of a database**

In this procedure, you will execute a script that lists all of the customers in the **Customers** table in order to evaluate the restore process.

- Open a query window, open Labfiles\L07>ListCust.sql, review its contents, and then execute it. This script determines whether the three new customers that were added to the **Customers** table in Exercise 2 were recovered.

Have all three new customers been recovered?

Yes.

Review

Topic Objective

To reinforce module objectives by reviewing key points.

Lead-in

The review questions cover some of the key concepts taught in the module.

- SQL Server Recovery Process
- Preparing to Restore a Database
- Restoring Backups
- Restoring Databases from Different Backup Types
- Restoring Damaged System Databases

*****ILLEGAL FOR NON-TRAINER USE*****

1. What is the automatic recovery process, and when is it initiated?

Automatic recovery occurs whenever SQL Server is restarted. It rolls transactions back or forward to maintain database integrity after a system failure.

2. What steps should you take before you restore a database?

Set the database to members of db_owner, dbcreator, or sysadmin. Then, back up the transaction log so that it may be applied at the completion of the restore operation.

3. You have a full database backup and several transaction log backups. Your database is spread among four files. The disk on which the third file resides fails. What should you do to restore and recover the database?

Restrict database access to members of db_owner, dbcreator, or sysadmin. Back up the transaction log so that it can be applied at the completion of the restore operation. Replace or fix the disk media. Restore the third backup file with the full backup as the source. Restore all but the last of the transaction log backups and specify the NORECOVERY option. Restore the last transaction log backup and specify the RECOVERY option.

-
4. Your database is set for the Full Recovery model. You have a full database backup and several transaction log backups. A malicious update to the database occurs at 9:21 A.M. The time is now 9:30 A.M. What should you do to restore and recover the database to a consistent state?

Back up the transaction log. Restrict database access to members of db_owner, dbcreator, or sysadmin. Restore the database, specifying the NORECOVERY option. Apply all but the last transaction log with the NORECOVERY option. Apply the last transaction log, specifying RECOVERY, STOPAT = 'month, xx, year, time' where the time is 9:20 A.M.

5. In the scenario that is presented in Question 4, will any changes be lost due to the restore process?

If any activity occurred between 9:20 and 9:30 A.M., these changes are lost.

