MICROSOFT
TRAINING
AND CERTIFICATION

Microsoft Official
Curriculum

Module 3: Managing Database Files

Contents

Microsoft

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:
90 Minutes

Lab:
30 Minutes

This module provides students with the knowledge and skills that they need to create a database. It describes how Microsoft® SQL Server™ 2000 stores data and processes transactions; how to create, manage, and place databases files and transaction logs; and how to optimize databases by using hardware-based RAID, user-defined filegroups, and a combination of both. It concludes with information on how to allocate, manage, and monitor the space and storage requirements for a database, as well as some performance considerations.

In the lab, students will create a database by using the Create Database Wizard and then create and modify a database by using SQL Server Enterprise Manager and Transact-SQL statements. Students will view and change database options by using Transact-SQL statements, and delete a database.

After completing this module, students will be able to:

- Describe how SQL Server stores data and handles transactions.
- Create a database, including specifying options during and after database creation.
- Grow, shrink, or delete a database.
- Determine the placement of database files and transaction logs for performance and fault tolerance.
- Optimize a database by using hardware-based RAID.
- Determine when and how to use filegroups to optimize a database.
- Optimize a database by using user-defined filegroups with hardware-based RAID.
- Estimate the amount of space that a database requires.

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 Microsoft PowerPoint® file 2072A_03.ppt.

Preparation Tasks

To prepare for this module, you should:

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

Multimedia Presentation

This section provides multimedia presentation procedures that do not fit in the margin notes or are not appropriate for the student notes.

Transactions

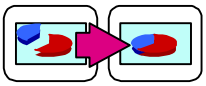
You must play the multimedia presentation in class because the content is not presented anywhere else in the module.

► **To prepare for the multimedia presentation**

- Click the button in the slide to start the multimedia presentation.

  This multimedia presentation shows how transactions work.

Other Activities



Displaying the Animated PowerPoint Slide

The animated slide is identified with an icon of links on the lower left corner of the slide.

► **To display the How the Transaction Log Works slide**

1. Display the topic slide, which shows the first step where the application sends the data modification.
2. Advance to the next animation, where the next step shows how affected data pages are loaded from disk into memory (called the buffer cache).

   Explain that affected data pages are loaded from disk into memory, provided that the pages are not already in the buffer cache from a previous query.
3. Advance to the next animation, where each data modification statement is recorded in the transaction log as it is made.

   Explain that the change is always recorded in the transaction log and written to disk before that change is made in the database. Mention that this type of log is called a write-ahead log.
4. Advance to the next animation, where the next step shows the checkpoint process writing all completed transactions to the database on the disk.

   Explain that this occurs on a recurring basis.

Module Strategy

Use the following strategy to present this module:

- Introduction to Data Structures

  Explain the importance of understanding how SQL Server stores data in order for students to allocate the appropriate disk space for data files and transaction logs. Then, play the multimedia presentation that describes how SQL Server handles and stores transactions.

- Creating Databases

  Describe what occurs during the process of creating a database and discuss the options that students choose during database creation. Then, explain how to change database options by using SQL Server Enterprise Manager and Transact SQL. Explain how to view database information and properties.

- Managing Databases

  Describe the different ways to manage data and log file growth: using automatic file growth, expanding file size manually, or creating secondary files. Also describe how to shrink databases and files automatically and manually, and mention the configuration options. Then, discuss how to delete a database, as well as restrictions on deleting databases.

- Placing Database Files and Logs

  Discuss how managing the placement of data files and transaction logs on disks can improve performance and implement fault tolerance. Keep the discussion general, because the following sections provide in-depth information.

- Optimizing a Database Using Hardware-based RAID

  Introduce how hardware-based RAID allows the management of multiple disks as one disk, and explain how students can implement this strategy in their own environments.

- Optimizing a Database Using Filegroups

  Present an overview of the concept of filegroups, and then describe the types of filegroups and how to size the default filegroup. Review the system stored procedures that display information about filegroups.

- Optimizing the Database Using Filegroups with Hardware-based RAID

  Describe how combining filegroups with hardware-based RAID solutions reduces contention and offers system or database administrators an approach that can be easily set up and managed.

- Capacity Planning

  Discuss the thought processes involved with planning the size of databases and associated files.

Customization Information

This section identifies the lab setup requirements for a module and the configuration changes that occur on student computers during the labs. 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 be 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 creating and managing databases and filegroups, using RAID and filegroups to optimize SQL Server, and how SQL Server stores data.

- **Introduction to Data Structures**
- **Creating Databases**
- **Managing Databases**
- **Placing Database Files and Logs**
- **Optimizing a Database Using Hardware-based RAID**
- **Optimizing a Database Using Filegroups**
- **Optimizing the Database Using Filegroups with Hardware-based RAID**
- **Capacity Planning**

This module describes how to create a database, set database options, create filegroups, RAID and filegroups to optimize Microsoft® SQL Server™ 2000, and manage a database and the transaction log.

After completing this module, you will be able to:

- Describe how SQL Server stores data and handles transactions.
- Create a database, including specifying options during and after database creation.
- Grow, shrink, or delete a database.
- Determine the placement of database files and transaction logs for performance and fault tolerance.
- Optimize a database by using hardware-based RAID.
- Determine when and how to use filegroups to optimize a database.
- Optimize a database by using filegroups with hardware-based RAID.
- Estimate the amount of space that a database requires.

Introduction to Data Structures

Topic Objective
To describe data structures.

Lead-in
When you create a database, you set up the data storage structure.

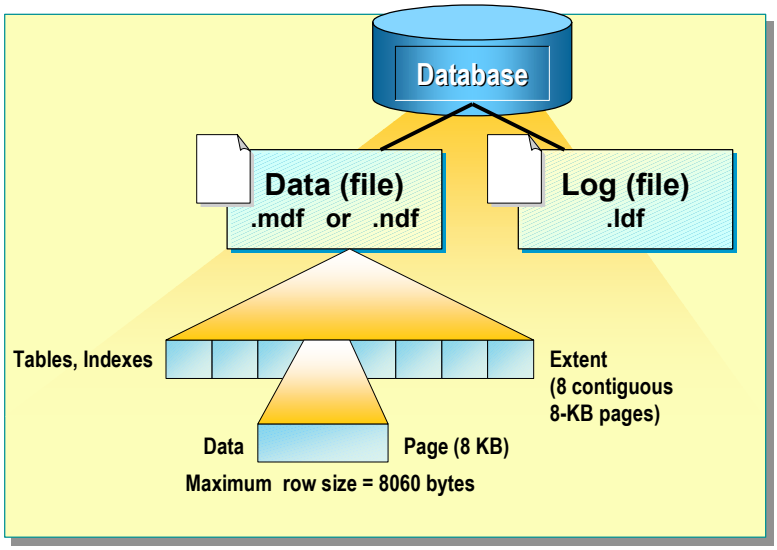
- How Data Is Stored
- Transactions
- How the Transaction Log Works

When you create a database, you set up the data storage structure. This structure includes at least one data file, which contains database objects, and one transaction log file. Before you create a database, it is important that you understand how SQL Server stores data, as well as the function of the transaction log.

How Data Is Stored

Topic Objective
To describe how the database is structured.

Lead-in
When you create a database, it is important to understand how SQL Server stores data.



Delivery Tip
These numbers are significant because they specify the maximum size of rows and extents; this is important to know when designing a database or capacity planning.

When creating a database, it is important to understand how SQL Server stores data so that you can calculate and specify the amount of disk space to allocate for the data files and transaction logs. Consider the following facts and guidelines about data storage:

- All databases have a primary data file (.mdf) and one or more transaction log files (.ldf). A database also may have secondary data files (.ndf). These physical files have both operating system file names and logical file names that can be used in Transact-SQL statements. The default location for all data files and transaction logs is C:\Program Files\Microsoft SQL Server\MSSQL\Data.
- When you create a database, a copy of the **model** database, which includes the system tables, is copied to the database.
- Data is stored in 8-kilobyte (KB) blocks of contiguous disk space called pages. This means that a database can store 128 pages per megabyte (MB).

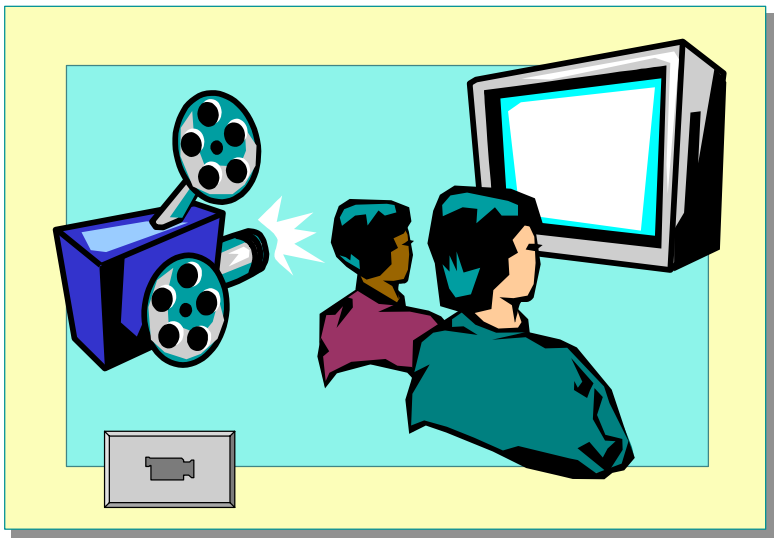
Delivery Tip
The size of a row is an important consideration when estimating the size of a database.

- Rows cannot span pages. Thus, the maximum amount of data in a single row, subtracting the space required for row overhead, is 8060 bytes.
- Tables and indexes are stored in extents. An extent is eight contiguous pages, or 64 KB. Therefore, a database has 16 extents per megabyte. Small tables can share extents with other database objects.
- Transaction log files hold the information necessary for recovery of the database in the event of a system failure.

Multimedia Presentation: Transactions

Topic Objective
To introduce the concept of how transactions operate.

Lead-in
Let's watch a multimedia presentation of how transactions work…



The main points of the multimedia presentation include:

Defining a Transaction

A transaction is a set of one or more Transact-SQL statements that are treated as a single unit of work and recovery. The unit must execute entirely, or not at all. Applications control transactions when you specify the beginning and end of transactions. You can use either Transact-SQL statements or database application programming interface (API) functions to specify the beginning and end of transactions. SQL Server performs implicit and explicit transactions.

Implicit Transaction

SQL Server performs an implicit transaction when any of the following Transact-SQL statements are executed as a transaction.

- ALTER TABLE
- CREATE
- DELETE
- DROP
- FETCH
- GRANT
- INSERT
- OPEN
- REVOKE
- SELECT
- TRUNCATE TABLE
- UPDATE

By default, SQL Server operates in autocommit mode. This means that an implicit transaction commits after execution without a COMMIT TRANSACTION statement to end the transaction.

SQL Server can also function in implicit transaction mode. This means that when any of the above Transact-SQL statements begins a transaction, the transaction must have a COMMIT TRANSACTION statement to end the transaction.

Explicit Transaction

SQL Server performs an explicit transaction when the beginning and end of the transaction are explicitly defined. You can define the beginning and end of the transaction in Transact-SQL by using BEGIN TRANSACTION and COMMIT TRANSACTION statements.

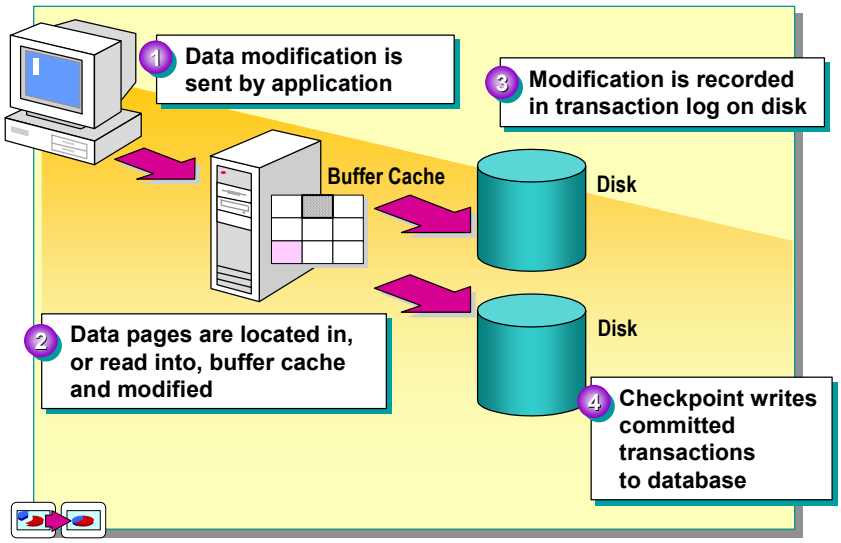
Explaining the Transaction Log

SQL Server records every transaction in a transaction log to maintain database consistency and aid in recovery. The log is a storage area that automatically tracks changes to a database. SQL Server records modifications in the log on disk as the modifications are executed, before they are written in the database.

How the Transaction Log Works

Topic Objective
To describe how the transaction log works.

Lead-in
The transaction log records data modifications as they occur.



Delivery Tip
The illustration consists of three animated images from which to teach.

The images reflect the steps mentioned in the student workbook.

Use the information in the Instructor Notes if you require additional teaching suggestions.

The transaction log records data modifications as they occur. The logging process is as follows:

1. A data modification is sent by the application.
2. When a modification is executed, the affected data pages are loaded from disk into memory (called the buffer cache), provided that the pages are not already in the buffer cache from a previous query.
3. Each data modification statement is recorded in the log as it is made. The change is always recorded in the log and written to disk before that change is made in the database. This type of log is called a write-ahead log.
4. On a recurring basis, the checkpoint process writes all completed transactions to the database on the disk.

If the system fails, the automatic recovery process uses the transaction log to roll forward all committed transactions and roll back any incomplete transactions.

Transaction markers in the log are used during automatic recovery to determine the starting and ending points of a transaction. A transaction is considered complete when the BEGIN TRANSACTION marker has an associated COMMIT TRANSACTION marker. Data pages are written to the disk when a checkpoint occurs.

Creating Databases

Topic Objective
To list the topics in this section.

Lead-in
When creating a database, you should understand what occurs during database creation, the types of options that you have during the process, what options you can change after you create the database, and how to view the database properties.

- What Occurs During Database Creation
- Specifying Options During Database Creation
- Changing Database Options After Database Creation
- Viewing Database Properties

When creating a database, you should understand what occurs during database creation, the types of options that you have during the process, what options you can change after you create the database, and how to view the database properties.

What Occurs During Database Creation

Topic Objective
To describe what occurs during database creation.

Lead-in
You can create a database by using SQL Server Enterprise Manager, or by executing Transact-SQL statements. When you create a database, SQL Server…

- **Creates a Data File and a Transaction Log**
- **Requires That the Owner and Creator Have Permission to the master Database**
- **Allows You to Define:**
  - The name of the database
  - The properties of the database
  - The location of the database files

Delivery Tip
Demonstrate the two ways to create a database by using SQL Server Enterprise Manager.

You can create a database by using SQL Server Enterprise Manager or the CREATE DATABASE Transact-SQL statement in SQL Query Analyzer.

When you create a database, SQL Server:

- Creates a data file and a transaction log for that database.
- Requires that the owner and creator of the new database have permission to use the **master** database, because information about each database in SQL Server is recorded in the **sysdatabases** and **sysaltfiles** tables in the **master** database.
- Allows you to define:
  - The name of the database.
  - The properties of the database.
  - The location of the database files.
- SQL Server uses a copy of the **model** database to initialize the database and its metadata. Any options or settings in the **model** database are copied into the new database.
- SQL Server then fills the rest of the database with empty pages, except for pages that have internal data recording how the space is used in the database.

Important You should back up the **master** database each time that you create, modify, or delete a database.

Specifying Options During Database Creation

Topic Objective
To describe which options are available during database creation.

Lead-in
When you create a database, you can specify the following options.

- Primary File
- Secondary Files
- Transaction Log
- File Name and Location
- Size
- File Growth
- Maximum Size
- Collation

When you create a database, you can specify the following options.

Primary File

The primary file consists of the initial data file in the primary filegroup. A *filegroup* is a named collection of data files. The primary filegroup contains all of the database system tables. It also contains all objects and data not assigned to user-defined filegroups. The primary data file is the starting point of the database and points to the rest of the files in the database. Every database has one primary data file and one primary filegroup. The recommended file name extension for primary data files is .mdf.

Secondary Files

Databases may have secondary data files. Some databases may be large enough to need multiple secondary data files, or they may use secondary files on separate disk drives to spread the data across multiple disks. Secondary files may be either in the primary filegroup or can be grouped into user-defined filegroups. The recommended file name extension for secondary data files is .ndf.

Transaction Log

Every database must have a transaction log. Unless specified otherwise, a transaction log file is automatically created with a system-generated name. The recommended file name extension for transaction log files is .ldf. Typically, the transaction log file is about 10 to 15 percent of the database files.

File Name and Location

Every database file has a logical name and a physical location for the file. Files generally should be spread across multiple disks for performance and redundancy.

Size

You can specify sizes for each data and log file. The minimum size is 512 KB for both the data or log file. The size specified for the primary data file must be at least as large as the primary file of the **model** database.

File Growth

You can specify whether a file will grow in size if necessary. This option is referred to as *autogrow*. The default is to enable file growth.

Delivery Tip
You can specify file growth in megabytes or as a percentage. The percentage only applies to file growth, not maximum size.

Maximum Size

You can specify the maximum size to which a file can grow in megabytes or as a percentage. The default growth value is 10 percent. It is recommended that you specify a maximum size to which the file is permitted to grow. If you do not specify a size, and file growth is enabled, by default the file will grow until the disk is full.

Delivery Tip
Mention that the file path will vary if a named instance of SQL Server is used instead of a default instance.

Collation

This parameter specifies the default collation for the database. By default, a database inherits the collation of the instance of SQL Server in which the database is created.

Changing Database Options After Database Creation

Topic Objective
To list the configurable database options.

Lead-in
After you have created a database, you can change the database options. You can configure a number of database options for each database.

Category of database options	Controls
Auto	Certain automatic behaviors
Cursor	Cursor behavior and scope
Recovery	The recovery model for the database
SQL	Control ANSI compliance options
State	• Whether the database is online or offline • Who can connect to the database • Whether the database is in read-only mode

After you have created a database, you can change the database options. You set database options by using SQL Server Enterprise Manager, the ALTER DATABASE Transact-SQL statement, or the **sp_dboption** stored procedure.

You can configure a number of database options for each database. To affect options in all new databases, you must change the **model** database.

There are five categories of database options:

- Auto options control certain automatic behaviors.
- Cursor options control cursor behavior and scope.
- Recovery options control the recovery model for the database.
- SQL options control ANSI compliance options.
- State options control:
  - Whether the database is online or offline.
  - Who can connect to the database.
  - Whether the database is in read-only mode.

Viewing Database Properties

Topic Objective
To discuss tools to view database properties.

Lead-in
To obtain database properties and information, you can…

- **Use SQL Server Enterprise Manager**
- **Use SQL Query Analyzer**
  - System functions
  - System stored procedures (**sp_helpdb** or **sp_spaceused**)
  - DBCC statements (DBCC SQLPERF (LOGSPACE))

Delivery Tip
Demonstrate how to view database information in SQL Server Enterprise Manager and SQL Query Analyzer.

To obtain database properties and information, you can use SQL Server Enterprise Manager, or system functions, system stored procedures, or Database Consistency Checker (DBCC) statements in SQL Query Analyzer.

The following table lists the commonly used procedures and statements.

System stored procedure / DBCC statement	Description
sp_helpdb	Reports on all databases on a server. Provides database name, size, owner, ID, creation date, and options.
sp_helpdb *database_name*	Reports on a specified database only. Provides database name, size, owner, ID, creation date, and options. Also lists files for data and log.
sp_spaceused [*objname*]	Summarizes the storage space that a database or database object uses.
DBCC SQLPERF (LOGSPACE)	Provides statistics about the use of transaction log space in all databases.

Managing Databases

Topic Objective
To list topics in this section.

Lead-in
In this section you'll learn how to manage databases and transaction logs.

- Managing Data and Log File Growth
- Shrinking a Database or Database File Automatically
- Shrinking a Database or Database File Manually
- Deleting a Database

As your database grows or changes, you can expand or shrink the size of data and log files manually, or configure them to expand or shrink automatically. When you no longer need a database, you can delete it, along with all associated files.

Managing Data and Log File Growth

Topic Objective
To explain how to manage data and transaction log file growth.

Lead-in
When data files grow, or when data modification activity increases, you may need to expand the size of the data or log files.

- **Using Automatic File Growth**
  - Specify space allocated, maximum size, and growth increment of each file
  - Optimize performance by allocating sufficient space, setting maximum size, and setting growth increments
- **Manually Expanding Data and Transaction Log Files**
- **Determining Whether to Expand Files Automatically or Manually**
- **Creating Secondary Data and Transaction Log Files**

When data files grow, or when data modification activity increases, you may need to expand the size of the data or log files. You can manage database growth by using SQL Server Enterprise Manager or the ALTER DATABASE statement. You must be in the **master** database to use the ALTER DATABASE statement.

You can control the size of the database by configuring the database and log files to grow automatically, manually increasing the existing database and log files, or creating secondary database and log files.

Using Automatic File Growth

You can set the automatic file growth property of any database file to specify that the file automatically expand by a specified amount or percentage whenever necessary. Using automatic file growth reduces the administrative tasks involved with manually increasing the database size.

You can specify the space allocated, maximum size, and growth increment of each file. If you do not specify a maximum size, a file can continue to grow until it uses all available space on the disk.

For optimum performance, you should:

- Allocate sufficient space to the database and the log to avoid frequently activating automatic growth.
- Set a maximum size for data files.
- Set the data and log file growth increments to sufficient sizes to avoid frequently activating automatic growth.

Manually Expanding Data and Transaction Log Files

You can also manually increase the size of any data or transaction log file by using either SQL Server Enterprise Manager or the ALTER DATABASE statement in Transact-SQL. You may want to manually expand database files to control when the expansion occurs. Expanding files in small increments increases fragmentation and can affect performance if files are expanded while the database is busy.

Determining Whether to Expand Files Automatically or Manually

Because you can set database files to grow automatically or expand files manually, you should consider the following:

- In a large production environment, you should allocate sufficient space for the data files you create the database and manually expand the files, if necessary. This allows you to control when to expand the files.
- In a desktop or small production environment, such as a sales people out in the field, setting database files to grow automatically reduces administrative overhead.

Creating Secondary Data and Transaction Log Files

You can also create secondary data and transaction log files to expand the size of a database. Use secondary data and transaction log files to place files on separate physical disks. You can also use RAID to spread data across multiple disks.

Shrinking a Database or Database File Automatically

Topic Objective
To discuss shrinking database or database files automatically.

Lead-in
When too much space is allocated, or when space requirements decrease, you can shrink an entire database or specific data or transaction log files.

- **Enabling Shrinking to Occur Automatically**
  - Specifying the **autoshrink** option in SQL Server Enterprise Manager
  - Executing the ALTER DATABASE AUTO_SHRINK statement
  - Executing the **sp_dboption** system stored procedure
- **SQL Server Activities During Auto Shrink**
  - Shrinks data and transaction log files when more than 25 percent of the file contains unused space
  - Performs shrinking in the background and does not affect user activity

When too much space is allocated, or when space requirements decrease, you can shrink an entire database or specific data or transaction log files.

Enabling Shrinking to Occur Automatically

You can set a database or database files to shrink automatically by:

- Specifying the **autoshrink** option in SQL Server Enterprise Manager.
- Executing the ALTER DATABASE AUTO_SHRINK statement.
- Executing the **sp_dboption** system stored procedure.

SQL Server Activities During Auto Shrink

By default, the option to shrink files automatically is disabled in all SQL Server editions, except the Desktop Edition. When you enable SQL Server to automatically shrink files, SQL Server:

- Shrinks data and transaction log files when more than 25 percent of the file contains unused space.

  The percentage of free space to be removed cannot be configured. SQL Server removes as much free space as possible. The transaction log is only shrunk if it does not contain active portions of the logical log that are required for database restoration.
- Performs this activity in the background and does not affect any user activity within the database.

Shrinking a Database or Database File Manually

Topic Objective
To discuss shrinking databases or database files manually.

Lead-in
Shrinking an entire database or individual database files manually is useful in a production environment because it allows you to control when to perform this activity

- **Methods of Shrinking**
- **Shrinking a Database and Data Files**
- **Shrinking Transaction Log Files**
  - Shrinks inactive portions of the transaction log that are larger than the desired size
  - If that is not enough to reduce to the desired size, SQL Server returns a message and notifies you what to perform
- **Configuring Shrink Database Options**

Shrinking an entire database or individual database files manually is useful in a production environment, because you have control over when the activity takes place.

Methods of Shrinking

To manually shrink databases and database files to a specific size, you can use SQL Server Enterprise Manager or execute the DBCC SHRINKDATABASE or DBCC SHRINKFILE statement. You can:

- Shrink data and transaction files as a group or individually.
- Shrink individual data and transaction log files that are smaller than their initial creation size by using the DBCC SHRINKFILE statement.

Shrinking a Database and Data Files

SQL Server shrinks a database and individual data files immediately. You cannot shrink an entire database to a size smaller than its initial creation size or that of the **model** database.

Shrinking Transaction Log Files

When you manually shrink transaction log files, SQL Server attempts to shrink the transaction log immediately. SQL Server:

1. Shrinks inactive portions of the transaction log that are larger than the desired size.
2. If that is not enough to reduce the transaction log to the desired size, SQL Server:
   a. Returns an informational message that states that part of the active log is beyond the desired size.
   b. Notifies you what to do (such as a back up the database to truncate the log) in order to move the active log from the end of the transaction log file. Moving the active log from the end allows SQL Server to shrink the transaction log file.

Delivery Tip
Emphasize the fact that the DBCC SHRINKDATABASE or DBCC SHRINKFILE statement must be re-executed to restart the shrink process.

Upon receipt of the messages from SQL Server, you can perform the suggested activity and then re-execute the DBCC SHRINKDATABASE or DBCC SHRINKFILE statement to complete the shrink process.

Configuring Shrink Database Options

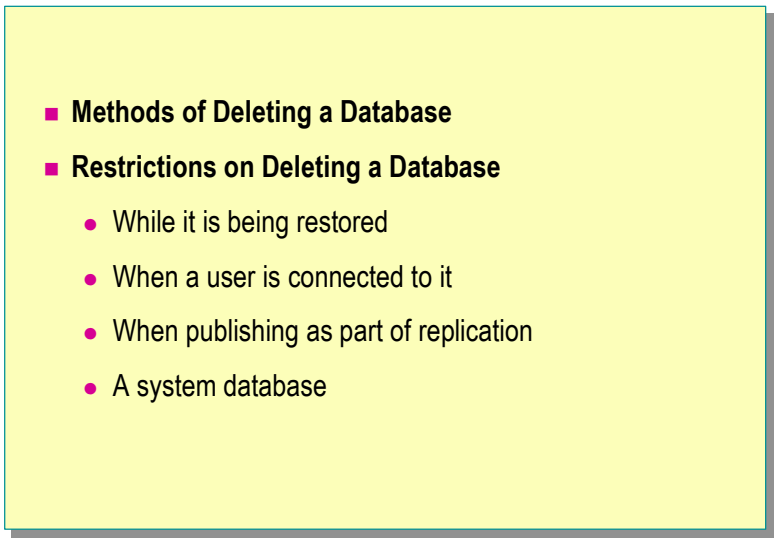
You can use the following database options when shrinking database files by using SQL Server Enterprise Manager.

Option	Description
Maximum free space in files after shrinking	The desired percentage of free space left in the database file after SQL Server has shrunk the database.
Move pages to beginning of file before shrinking	Moves pages to the beginning of the file before shrinking the database. Selecting this option may affect performance but may be necessary to meet the desired shrinkage goal.
Shrink the database based on this schedule	Shrinks the database on a selected schedule.
Files	Specifies the individual database files to shrink. This provides more precise control when shrinking the database.

Deleting a Database

Topic Objective
To illustrate how to delete a database.

Lead-in
Delete a database only when you are certain that it is no longer needed.



You can delete a database when you no longer need it. Deleting a database removes the database and the disk files that the database uses.

Methods of Deleting a Database

You can delete databases by using SQL Server Enterprise Manager or by executing the DROP DATABASE statement. After you delete a database, every login ID that used that particular database as its default database will not have a default database.

Note Back up the **master** database after you delete a database.

Restrictions on Deleting a Database

The following restrictions apply to deleting databases. You cannot delete:

- A database that is in the process of being restored.
- A database that is open for reading or writing by any user.
- A database that is publishing any of its tables as part of SQL Server replication.
- A system database.

Placing Database Files and Logs

Topic Objective
To discuss placing database files and transaction logs appropriately.

Lead-in
You can improve performance and implement fault tolerance by managing the placement of data files and transaction logs on disks.

- **Managing Disk Storage**
  - Performance
  - Fault tolerance
- **Spreading Data Files**
- **Creating Transaction Logs on Separate Disks**
- **Placing the tempdb Database**

You can improve performance and implement fault tolerance by managing the placement of data files and transaction logs on disks.

SQL Server uses Microsoft Windows® 2000 input/output (I/O) calls to perform disk reads and writes. SQL Server manages when and how disk I/O is performed but relies on Windows to perform the underlying I/O operations. The I/O subsystem includes the system bus, disk controller cards, disks, tape drives, CD-ROM drives, and many other I/O devices. The disks are frequently the biggest bottleneck in a system.

Managing Disk Storage

In the context of managing disk storage for SQL Server,

Performance refers in part to the speed of read and write operations.

Fault tolerance refers to the ability of the system to continue functioning without data loss when part of the system fails.

In general, use disks formatted with NTFS that use 64 KB as the allocation unit. Do not use compressed volumes.

Spreading Data Files

You should spread as much data across as many physical drives as possible. Doing so improves throughput through parallel data access by using multiple files. In general, create one file for each physical disk and group the files into one or more filegroups. SQL Server can perform:

- Parallel scans of the data if the computer has multiple processors and multiple disks.
- Multiple parallel scans for a single table if the filegroup of the table contains multiple files.

To spread data evenly across all disks, use RAID, and then use user-defined filegroups to spread data across multiple hardware stripe sets, if needed.

Note An advanced technique is to separate the tables from *nonclustered* indexes. A nonclustered index is an index in which the logical order of the index is different than the physical, stored order of the rows on disk.

Creating Transaction Logs on Separate Disks

You should create the transaction log on a separate disk, or use RAID. Because the transaction log file is written serially, using a separate, dedicated disk allows the disk heads to stay in place for the next write operation. Using RAID provides fault tolerance.

For example, if your production environment has multiple databases on a server, you might want to use separate disks for each transaction log. This strategy allows optimal performance.

Placing the tempdb Database

You should place the **tempdb** database on a fast I/O subsystem separate from user databases to ensure optimal performance. You can use RAID to stripe the **tempdb** database across multiple disks for better performance.

Optimizing a Database Using Hardware-based RAID

Topic Objective
To discuss how to optimize a database by using hardware-based RAID.

Lead-in
Hardware-based RAID allows you to manage multiple disks by handling an array of disks as one disk.

- **Using Hardware-based RAID**
  - Offers better performance than operating system-based RAID
  - Enables you to replace failed drive without shutting down the system
- **Applying Types of RAID**
  - Disk mirroring or disk duplexing (RAID 1) for redundancy for the transaction log
  - Disk striping with parity for performance and redundancy for data files and transaction logs
  - Disk mirroring with striping for maximum performance for data files

Hardware-based RAID allows you to manage multiple disks by handling an array of disks as one disk.

Using Hardware-based RAID

You should use hardware-based RAID rather than operating system-based RAID for better performance. Using operating system-based RAID takes CPU cycles away from other system requirements. Using hardware-based RAID might also enable you to replace a failed drive without shutting down the system. However, this advantage depends upon the particular implementation of hardware RAID that you purchase.

Applying Types of RAID

When using hardware–based RAID to optimize your database, consider using the following types of RAID:

- Disk mirroring or disk duplexing (RAID 1) for redundancy for the transaction log.
- Disk striping with parity (RAID 5) for performance and redundancy for data files and transaction logs.
- Disk mirroring with striping (RAID 10 or RAID 1 + RAID 0) for maximum performance for data files.

Important Using RAID for fault tolerance does not replace proper backup strategies. You must back up periodically to protect your databases and data against catastrophic loss.

Optimizing a Database Using Filegroups

Topic Objective
To introduce the topics in this section.

Lead-in
In this section, you'll learn how to use filegroups to improve performance.

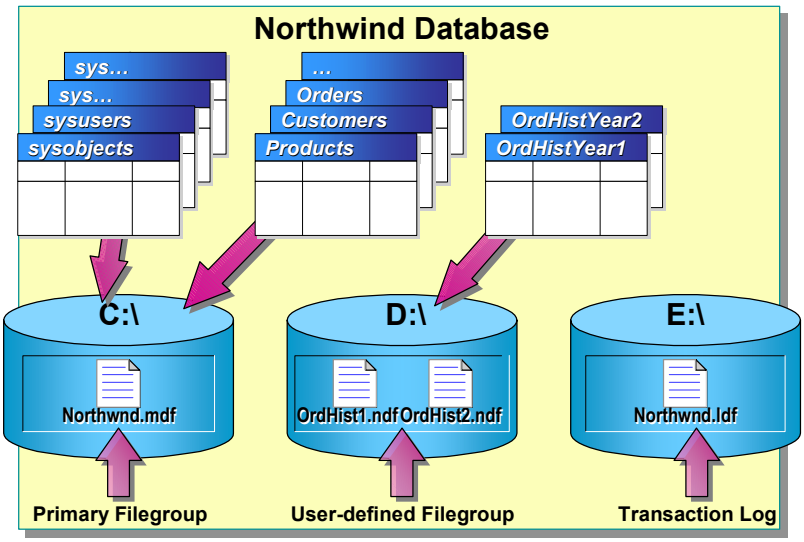
- Introduction to User-defined Filegroups
- Creating User-defined Filegroups
- Using Filegroups for Performance
- Using Filegroups for Maintenance
- Considerations When Creating Filegroups

Filegroups improve performance by distributing data across multiple disks and by using parallel threads for query processing. Filegroups also can facilitate database maintenance.

Introduction to User-defined Filegroups

Topic Objective
To introduce the concept of filegroups.

Lead-in
If your hardware setup includes multiple disk drives, you can locate specific objects and files on individual disks, grouping your database files into one or more user-defined filegroups



If your hardware setup includes multiple disk drives, you can locate specific objects and files on individual disks, grouping your database files into one or more filegroups.

Types of Filegroups

SQL Server has a primary filegroup and may also have user-defined filegroups.

- The primary filegroup contains the primary data file with the system tables.
- A user-defined filegroup consists of data files that are grouped together for allocation and administrative purposes.

Delivery Tip
Explain that if the Ordhist1.ndf and Ordhst2.ndf are both heavily queried, they should be placed on separate disks.

Placing Files on Separate Disks

The illustration is an example of how you might place database files on separate disks.

- You could create user-defined filegroups to separate files that are heavily queried from those that are heavily modified. In the illustration, the OrdHist1.ndf and OrdHist2.ndf files are placed on a separate disk from the **Products**, **Customers**, and **Orders** tables because they are queried for decision support rather than updated with current order information.
- You could also place the Ordhist1.ndf and Ordhst2.ndf files on separate disks if they are both heavily queried.
- Transaction log files are not part of a filegroup. Transaction log space is managed separately from data space.

Creating User-defined Filegroups

Topic Objective
To discuss creating filegroups.

Lead-in
You can create multiple files on separate disks and create a user-defined filegroup to contain the files.

- **Methods of Creating User-defined Filegroups**
- **Choosing a Default Filegroup**
  - SQL Server designates one filegroup as the default filegroup
  - Default filegroup is set to primary filegroup
  - Change the primary default filegroup if you create user-defined filegroups
- **Sizing the Primary Default Filegroup**
- **Viewing Filegroup Information**

You can create multiple data files on separate disks and create a user-defined filegroup to contain the files. If you use user-defined filegroups, try to have one file per physical disk.

Methods of Creating User-defined Filegroups

You can create a user-defined filegroup at the time that you create a database, or at a later time. You can use SQL Server Enterprise Manager or either the CREATE DATABASE or ALTER DATABASE statement.

Choosing a Default Filegroup

SQL Server designates one filegroup as the default filegroup. The default filegroup is set to the primary filegroup at the time of database creation, unless you specify otherwise. The default filegroup contains the pages for all tables and indexes that do not have a filegroup specified when they are created.

Sizing the Primary Default Filegroup

If the default filegroup is left as the primary filegroup, you must size the primary filegroup appropriately or set it to automatically grow so that you do not run out of space. The primary filegroup must be large enough to hold all system tables and any tables and indexes not allocated to a user-defined filegroup.

If the primary filegroup runs out of space, you will be unable to add any information to the system tables. However, if a user-defined filegroup runs out of space, only the user files that are specifically allocated to that filegroup are affected.

Viewing Filegroup Information

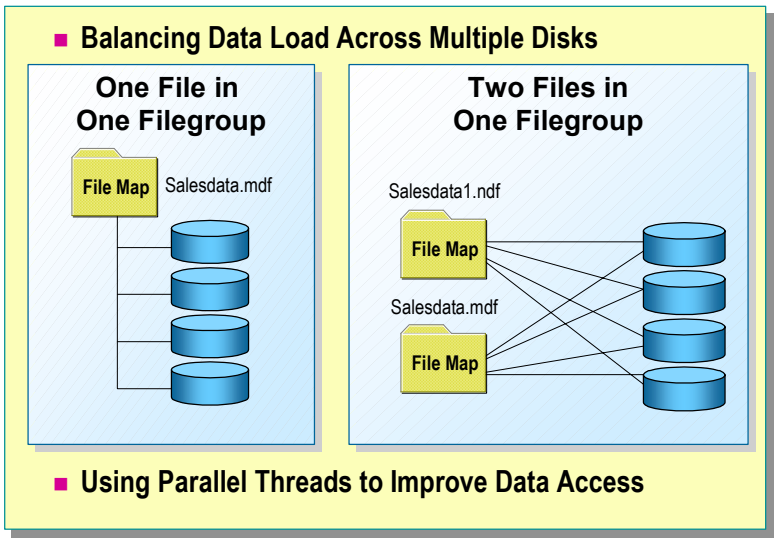
You can view information about filegroups by using SQL Server Enterprise Manager or system stored procedures in Transact-SQL.

System stored procedure	Description
sp_helpfile [[**@filename** =] '*name*']	Returns the physical names and attributes of files associated with the current database. Use this system stored procedure to determine the names of files to attach to or detach from the server.
sp_helpfilegroup *[filegroup_name]*	Returns the names and attributes of filegroups associated with the current database.

Using Filegroups for Performance

Topic Objective
To illustrate how to use filegroups to improve performance.

Lead-in
Filegroups improve performance by balancing data load across multiple disks and by using parallel threads to improve data access.



Using user-defined filegroups can improve performance by balancing data load across multiple disks and by using parallel threads to improve data access.

Balancing Data Load Across Multiple Disks

When you create a table, you can assign it to a user-defined filegroup. Filegroups use a proportional fill strategy across all files within the filegroup. As data is written to the filegroup, each file is filled in parallel.

Each file is physically placed on a disk or set of disks. SQL Server maintains a file map that associates each database object with its location on the disk. The illustration shows that:

- If one file is created on a filegroup that spans four disks, one file map points to the location of data on all four physical disks.
- If two files are created on a filegroup that spans four disks, two file maps (one map for each file) point to the location of the data on all four physical disks.

Using Parallel Threads to Improve Data Access

Whenever a table is accessed sequentially, the system creates a separate thread for each file in parallel. When the system performs a table scan for a table in a filegroup with four files, it uses four separate threads to read the data in parallel.

In general, using multiple files on separate disks improves performance. However, too many files in a filegroup can cause too many parallel threads and create bottlenecks.

Using Filegroups for Maintenance

Topic Objective
To describe how to use filegroups for maintenance.

Lead-in
You can use filegroups to facilitate maintenance.

- **Back Up or Restore Files or Filegroups Rather Than an Entire Database**
- **Group Tables and Indexes with Similar Maintenance Requirements into Same Filegroups**
- **Assign an Individual High-Maintenance Table to Its Own Filegroup**

In addition to using filegroups to balance data loads for performance, you can use filegroups to facilitate maintenance.

To use filegroups to simplify maintenance, you can:

- Back up or restore individual files or filegroups instead of backing up or restoring an entire database. Backing up files or filegroups may be necessary on large databases in order to have an effective backup and restore strategy.
- Group tables and indexes with similar maintenance requirements into the same filegroups.

  You may want to perform maintenance on some objects more frequently than others. For example, by creating two filegroups and assigning tables to them, you can run daily maintenance tasks against the tables in a daily group and weekly maintenance tasks against the tables in a weekly group. This limits disk contention between the two filegroups.
- Assign an individual high-maintenance table to its own filegroup.

  For example, a table that has frequent updates may need to be backed up and restored separately from the database as a whole.

Considerations When Creating Filegroups

Topic Objective
To discuss the effects of creating filegroups.

Lead-in
Creating user-defined filegroups is an advanced database design technique. When creating filegroups, you should…

- **Monitor System Performance**
- **Use Maintenance Requirements Rather Than Performance Considerations**
- **Specify a User-defined Filegroup as the Default**
- **Be Aware That Filegroups Do Not Provide Fault Tolerance**

Creating user-defined filegroups is an advanced database design technique. You must understand your database structure, data, transactions, and queries thoroughly in order to determine the best way to place tables and indexes on specific filegroups.

When creating filegroups, you should:

- Use maintenance requirements rather than performance considerations to determine the number of filegroups.

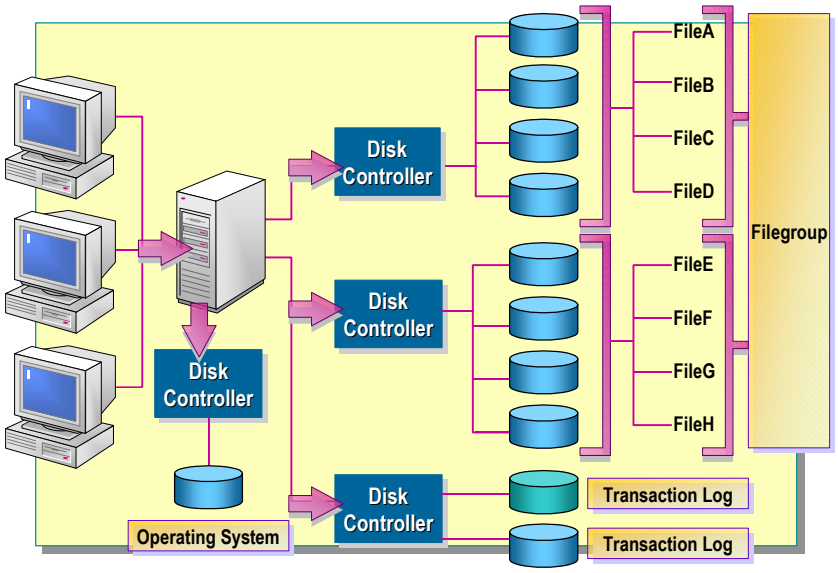
  In many cases, using the striping capabilities of RAID provides much of the same performance gain that you might achieve by using user-defined filegroups without the added administrative burden of defining and managing them.
- Change the default filegroup if you use user-defined filegroups. If your database has multiple filegroups, you should assign one of the user-defined filegroups as the default. This will prevent unexpected table growth from constraining the system tables in the primary filegroup.
- Be aware that filegroups do not provide fault tolerance. To include fault tolerance, you can mirror each disk by using RAID 1. However, this is an expensive option.

Delivery Tip
Point out that it is possible to over-engineer a database. Applications may receive more benefits from RAID than from user-defined filegroups.

Optimizing the Database Using Filegroups with Hardware-based RAID

Topic Objective
To illustrate how filegroups can be combined with hardware-based RAID.

Lead-in
You can combine filegroups with hardware-based RAID solutions.



You can combine filegroups with hardware-based RAID solutions. First, set up hardware striping, and then use filegroups to spread data across multiple hardware stripe sets.

The illustration shows two controllers pointing to two hardware stripe sets. Four files are associated with each stripe set. A filegroup contains all files on both stripe sets. This option spreads the data evenly across all disks while keeping administration simple.

This configuration uses the best features of hardware-based RAID and filegroups. It provides parallel data access by using a separate thread for each file, and it spreads the load among multiple disks to reduce contention. Because this approach creates one logical grouping, it is easy for a system or database administrator to set up and manage.

◆ Capacity Planning

Topic Objective
To introduce thought processes involved with planning the size of databases and associated files.

Lead-in
When you estimate the size of your database and transaction log files, you should be aware of the following…

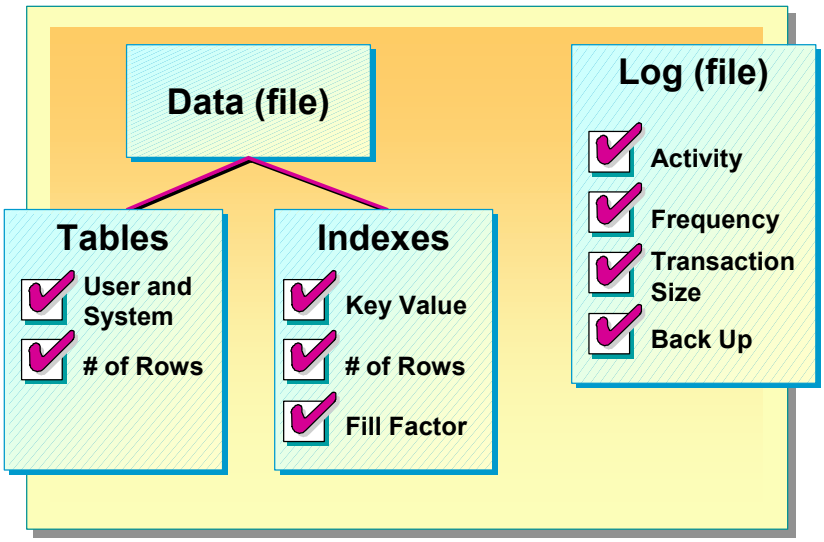
- **Estimating the Size of a Database**
- **Estimating the Amount of Data in Tables**

One of the main functions of a system or database administrator is to allocate, manage, and monitor the space and storage requirements for SQL Server and its databases. Estimating the space that a database requires can help you plan your storage layout and determine hardware requirements.

Estimating the Size of a Database

Topic Objective
To review the SQL Server physical objects that requires storage space.

Lead-in
When you estimate the size of your database, it is important to understand the factors that affect the size of the physical containers that store the data.



When you plan your database, you set up the logical structure. Underneath that structure are several physical files and objects that occupy disk space. These include the transaction log and the tables and indexes that make up the data files.

When you create a database, SQL Server creates a copy of the **model** database, including the system tables that contain information on files, objects, permissions, and constraints. These tables grow in size as you create objects in your database. Each object that you create generates a new row to be inserted into one or more system tables.

Factors to Consider When Estimating the Size of a Database

Consider the following factors when you estimate the amount of space that your database will occupy:

- Size of **model** database and system tables, including projected growth
- Amount of data in tables, including projected growth
- Number and size of indexes, especially the size of the key value, the number of rows, and the fill factor setting

- Size of the transaction log, which is influenced by the amount and frequency of modification activity, the size of each transaction, and how often you back up or dump the log
- Size of system tables, such as the number of users, objects, and so on, which typically is not a large percentage of the database size

Note As a starting point, you should allocate 10 to 25 percent of the database size to the transaction log for online transaction processing (OLTP) environments. You can allocate a smaller percentage for databases that are used primarily for queries.

Estimating the Amount of Data in Tables

Topic Objective
To familiarize students with the way that SQL Server stores information.

Lead-in
To estimate the amount of data that will be stored in your database tables, you first must understand the file structure that is used to create a database.

- **Calculate Number of Bytes in a Row**
  - Total the bytes in the row
  - Average variable-length columns
- **Determine Number of Rows in a Data Page**
  - Divide 8060 by the total bytes in the row
  - Round down to the next whole number
- **Divide Number of Rows in the Table by Number of Rows in a Data Page**

After you consider the amount of space that is allocated to the **model** database, you should estimate the amount of data in your tables, including projected growth. This can be calculated by determining the total number of rows, row size, number of rows that fit on a page, and the total number of pages that are required for each table in the database.

You can estimate the number of pages that are required for a table and the disk space that the table occupies if you know the number of characters for each row and the approximate number of rows that the table will have. Use the following method:

- Calculate the number of bytes in a row by totaling the number of bytes that each column contains. If one or more columns are defined as variable length—such as a column for names—you can add the column average to the total.
- Determine the number of rows that are contained in each data page. To do this, divide 8060 by the number of bytes in a row. Round the result down to the next whole number.
- Divide the approximate number of rows in the table by the number of rows that are contained in each data page. The result equals the number of pages that are needed to store your table.

Note A row cannot be larger than one page.

Performance Considerations

Topic Objective
To discuss some guidelines related to performance.

Lead-in
If you want to achieve the best performance from your database, consider the following guidelines.

- **Use RAID to Improve Performance or Fault Tolerance**
- **Place Data Files and Transaction Logs on Separate Physical Disks**
- **Use User-defined Filegroups to Simplify Backup Strategies of Very Large Databases**

If you want to achieve the best performance from your database, consider the following guidelines.

- Use RAID to improve performance and provide fault tolerance. You should use hardware-based RAID to gain faster access to data and to increase the safety of your data, and apply the appropriate RAID level to achieve the performance gains that you want while still maintaining the fault tolerance levels that you require.

  When possible, choose RAID disk striping before choosing user-defined filegroups.
- Place data files and transaction logs on separate physical disks with separate I/O controllers. This ensures that write operations to the transaction log do not compete with concurrent INSERT, UPDATE, or DELETE actions to the database tables.
- Use user-defined filegroups to place database objects on separate disks to simplify backup strategies of very large databases. This allows you to set individual backup strategies based on how often data is revised. If you have a group of files that changes often, you can back up those tables or objects frequently.

Recommended Practices

Topic Objective
To present recommended practices for creating and managing databases.

Lead-in
The following are recommended practices for creating and managing databases.

The following recommended practices will help you create and manage databases:

- Back up the **master** database immediately after you create or modify a database.
- Specify a maximum size when you use automatic file growth.

  This will prevent any one file from filling the entire hard disk.
- Make the initial database file size and autogrow increments large enough to avoid frequent file growth.

  This will reduce SQL Server administrative activity and help keep a file from becoming fragmented on the hard disk.
- Use disk mirroring, disk striping with parity, or disk mirroring with striping for performance and fault tolerance.
- Create one file for each physical disk and group them into a single primary filegroup.
- Change the default filegroup if you use user-defined filegroups. If your database has multiple filegroups, you should assign one of the user-defined filegroups as the default.

  This will prevent unexpected table growth from constraining the system tables in the primary filegroup.

Lab A: Managing Database Files

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will create and modify databases.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Create a database by using the Create Database Wizard.
- Create a database by using SQL Server Enterprise Manager or the CREATE DATABASE statement.
- Modify a database by using SQL Server Enterprise Manager or Transact-SQL statements.
- View and modify database options by using SQL Server Enterprise Manager or Transact-SQL statements.
- Delete a database by using SQL Server Enterprise Manager or TransactSQL-statements.

Prerequisites

Before working on this lab, you must have script files for this lab. These script files are located in C:\Moc\2072A\Labfiles\L03.

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: 30 minutes

Exercise 1
Using the Create Database Wizard

In this exercise, you will create a database by using the Create Database Wizard in SQL Server Enterprise Manager.

► To create a database by using the Create Database Wizard in SQL Server Enterprise Manager

In this procedure, you will use the Create Database Wizard in SQL Server Enterprise Manager to create a database.

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

Option	Value
User name	**SQLAdmin*x*** (where *x* corresponds to your computer name as designated in the nwtraders.msft classroom domain)
Password	**password**

2. Start SQL Server Enterprise Manager.
3. In the console tree, expand **Microsoft SQL Servers**, and then expand **SQL Server Group**.
4. Click your server.
5. On the **Tools** menu, click **Wizards**.
6. In the **Select Wizard** dialog box, expand **Database**, and then double-click **Create Database Wizard**.
7. Complete the wizard by using the information in the following table. Use the defaults provided by the wizard for any options that the table does not specify.

   Database name: **SampleDBWizard**

File	Location	File name	Initial size	Growth size	Maximum file size
Database	C:\Program Files\Microsoft SQL Server\MSSQL\Data	SampleDBWizard_Data	2 MB	2 MB	Unrestricted
Log	C:\Program Files\Microsoft SQL Server\MSSQL\Data	SampleDBWizard_Log	2 MB	1 MB	Unrestricted

   Do not create a maintenance plan at this time.

8. In the console tree, expand **Databases**, right-click **SampleDBWizard**, and then click **Properties**.

   Review the properties of the **SampleDBWizard** database to verify that the database has been created properly.

9. Close the **SampleDBWizard Properties** dialog box.

Exercise 2
Creating a Database

In this exercise, you will create a database by using SQL Server Enterprise Manager and Transact-SQL statements.

► To create a database by using SQL Server Enterprise Manager

In this procedure, you will create a database by using SQL Server Enterprise Manager.

1. In the SQL Server Enterprise Manager console tree, right-click **Databases**, and then click **New Database**.
2. Use the information in the following table to create a new database. Use the defaults for any options that the table does not specify.

   Database name: **SampleDBEM**

   Collation name: Use the server default

File	Location	File name	Initial size	Growth size	Maximum file size
Database	SampleDBEM_Data	C:\Program Files\Microsoft SQL Server\MSSQL\Data	5 MB	25%	15 MB
Log	SampleDBEM_Log	C:\Program Files\Microsoft SQL Server\MSSQL\Data	2 MB	50%	5 MB

3. View the database properties of the **SampleDBEM** database to verify that the database has been created properly.

► To create a database by using Transact-SQL statements in SQL Query Analyzer

In this procedure, you will use the CREATE DATABASE statement to create a database by using the information in the following table.

Database name**: SampleDBTsql**

File	File name	Location	Initial size	Maximum file size	Growth size
Database	SampleDBTsql_Data.mdf	C:\Program Files\Microsoft SQL Server\MSSQL\Data	7 MB	Unrestricted	3 MB
Log	SampleDBTsql_Log.ldf	C:\Program Files\Microsoft SQL Server\MSSQL\Data	3 MB	10 MB	1 MB

1. 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 **SQLAdmin*x***, 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.

2. Open C:\Moc\2072A\Labfiles\L03\Creasmpl.sql, review its contents, and then execute it.

   This script creates the **SampleDBTsql** database by using the CREATE DATABASE statement

3. Switch to SQL Server Enterprise Manager.
4. In the console tree, right-click **Databases**, and then click **Refresh**.
5. View the database properties of the **SampleDBTsql** database to verify that the database has been created properly.

Exercise 3
Modifying a Database

In this exercise, you will modify a database by changing the initial size, the growth increment, and the maximum size specification of a database file.

► To modify a database by using SQL Server Enterprise Manager

In this procedure, you will change the initial size, the growth increment and the maximum size of the **SampleDBEM** database that you created in Exercise 2.

1. In SQL Server Enterprise Manager, in the console tree, expand **Databases**, right-click **SampleDBEM**, and then click **Properties**.
2. Use the information in the following table to modify the properties of the **SampleDBEM** database.

File name	Space allocated	Growth increment	Maximum file size
SampleDBEM_Data	10 MB	10%	Unrestricted
SampleDBEM_Log	5 MB	20%	15 MB

3. View the database properties for the **SampleDBEM** database to verify that the database has been modified properly.

► To modify a database by using Transact-SQL statements

In this procedure, you will increase the maximum size of the log file to 20 megabytes (MB) for the **SampleDBTsql** database that you created in Exercise 2.

1. Switch to SQL Query Analyzer.
2. Open C:\Moc\2072A\Labfiles\L03\Modismpl.sql, review its contents, and then execute it.

   This script alters the **SampleDBTsql** database by using the ALTER DATABASE statement.
3. Switch to SQL Server Enterprise Manager and view the database properties for the **SampleDBTsql** database to verify that the database has been modified properly.

Exercise 4
Viewing Database Information and Changing Database Options

In this exercise, you will view information about databases and change database options.

► To view information about databases by using SQL Server Enterprise Manager

In this procedure, you will use SQL Server Enterprise Manager to view information about the **SampleDBEM** database.

1. In the SQL Server Enterprise Manager console tree, expand **Databases**, and then click **SampleDBEM**.
2. On the **View** menu, click **Taskpad**.
3. In the details pane, review the displayed information.

► To view information about databases by using Transact-SQL statements

In this procedure, you will use system stored procedures to view information about previously created databases.

1. Switch to SQL Query Analyzer and, on the toolbar, click **Clear Window**.
2. In the query window, type and then execute the following system stored procedure to display information about all databases in this instance of SQL Server.

   ```
   EXEC sp_helpdb
   ```

3. Type and then execute the following system stored procedure to display information about the **SampleDBTsql** database.

   ```
   EXEC sp_helpdb SampleDBTsql
   ```

4. On the toolbar, click **Clear Window**.
5. Type and then execute the following system stored procedure to display information about the amount of space used in the **SampleDBTsql** database.

   ```
   USE SampleDBTsql
   EXEC sp_spaceused
   ```

6. On the toolbar, click **Clear Window**.
7. Type and then execute the following system stored procedure to display information about space usage for the **sysobjects** table in the **SampleDBTsql** database.

   ```
   EXEC sp_spaceused sysobjects
   ```

► To change the database to read-only mode by using SQL Server Enterprise Manager

In this procedure, you will use SQL Server Enterprise Manager to change database options.

1. In the SQL Server Enterprise Manager console tree, expand **Databases**, right-click **SampleDBEM**, and then click **Properties**.
2. Click the **Options** tab, and review the available database options.
3. Select the **Read-only** check box to change the **SampleDBEM** database to read-only mode, and then click **OK**.

► To view database options and change database options by using Transact-SQL statements

In this procedure, you will use the **sp_dboption** system stored procedure to view database options and change the **SampleDBTsql** database to read-only mode.

1. Switch to SQL Query Analyzer and, on the toolbar, click **Clear Window**.
2. In the query window, type and then execute the following system stored procedure to view a list of settable database options.

   ```
   EXEC sp_dboption
   ```

3. Type and execute the following system stored procedure to view a list of database options that are enabled for the **SampleDBTsql** database.

   ```
   EXEC sp_dboption SampleDBTsql
   ```

4. Type and then execute the following **sp_dboption** system stored procedure to change the **SampleDBTsql** database to read-only mode.

   ```
   EXEC sp_dboption SampleDBTsql, 'read only', 'true'
   ```

5. On the toolbar, click **Clear Window**.
6. Type and then execute the following **sp_dboption** system stored procedure to verify that the **SampleDBTsql** database is now in read-only mode.

   ```
   EXEC sp_dboption SampleDBTsql
   ```

Exercise 5
Deleting a Database

In this exercise, you will delete a database by using SQL Server Enterprise Manager and Transact-SQL statements.

► To delete a database by using SQL Server Enterprise Manager

In this procedure, you will use SQL Server Enterprise Manager to delete the **SampleDBEM** database.

1. In the SQL Server Enterprise Manager console tree, expand **Databases**, right-click **SampleDBEM**, and then click **Delete**.
2. Click **Yes** to delete the **SampleDBEM** database and all backup and restore history for the database.
3. Verify that the **SampleDBEM** database has been deleted.

► To delete one or more databases by using Transact-SQL statements

In this procedure, you will use Transact-SQL statements to delete two databases that you created in a previous exercise.

1. Switch to SQL Query Analyzer.
2. Open C:\Moc\2072A\Labfiles\L03\Dropdb.sql, review its contents, and then execute it.

   This script deletes the **SampleDBTsql** and **SampleDBWizard** databases by using a single DROP DATABASE statement.
3. Review the output in the results pane and verify that you have deleted the **SampleDBTsql** and **SampleDBWizard** databases.

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.

- Introduction to Data Structures
- Creating Databases
- Managing Databases
- Placing Database Files and Logs
- Optimizing a Database Using Hardware-based RAID
- Optimizing a Database Using Filegroups
- Optimizing the Database Using Filegroups with Hardware-based RAID
- Capacity Planning

1. You are creating a database that is updated infrequently; it is used mainly for decision support and is query-intensive. What percentage of the database would you allocate to the transaction log?

   Because the database has little insert, update, or delete activity, it might make sense to reduce the default percentage from 25 percent to something closer to 10 percent. These percentages are general estimates only. The actual size would depend on the specific environment, including the size of the database, the size of the transactions involved, and the recovery model used.

2. What are the advantages of using user-defined filegroups? What is the main disadvantage?

   The advantages are that you can place tables on specific disks to eliminate disk contention and simplify backup strategies. The main disadvantage is the administrative complexity involved.

3. What is the advantage of using RAID?

   RAID provides fault tolerance, improves performance by spreading data across multiple drives, and is simpler to administer than user-defined filegroups.