MICROSOFT
TRAINING
AND CERTIFICATION

Microsoft Official Curriculum

Module 3: Creating and Managing Databases

Contents

Microsoft

Project Lead: Rich Rose
Instructional Designers: Rich Rose, Cheryl Hoople, Marilyn McGill
Instructional Software Design Engineers: Karl Dehmer, Carl Raebler, Rick Byham
Technical Lead: Karl Dehmer
Subject Matter Experts: Karl Dehmer, Carl Raebler, Rick Byham
Graphic Artist: Kirsten Larson (Independent Contractor)
Editing Manager: Lynette Skinner
Editor: Wendy Cleary
Copy Editor: Edward McKillop (S&T Consulting)
Production Manager: Miracle Davis
Production Coordinator: Jenny Boe
Production Support: Lori Walker (S&T Consulting)
Test Manager: Sid Benavente
Courseware Testing: TestingTesting123
Classroom Automation: Lorrin Smith-Bates
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, Data Base: Margo Crandall
Group Manager, Courseware Infrastructure: David Bramble
Group Product Manager, Content Development: Dean Murray
General Manager: Robert Stewart

Instructor Notes

Presentation:
30 Minutes

Lab:
30 Minutes

This module provides students with a description of how to create a database, set database options, create filegroups, and manage a database and the transaction log. It reviews disk space allocation and how the transaction log records data modifications.

Note This course is based on the **Northwind** database. The schema for the **Northwind** database is in Appendix A. The labs use a parallel version of the **Northwind** database that is called **ClassNorthwind**.

In the lab, students define the **ClassNorthwind** database, modify it, and then set an option to clear the transaction log.

After completing this module, students will be able to:

- Create a database.
- Create a filegroup.
- Manage a database.
- Describe data structures.

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 2073A_03.ppt
- The C:\Moc\2073A\Demo\D03_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.
- Practice demonstrating the animated slide.

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 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.

► **To display the Pages That Track Tables and Indexes slide**

This animated slide shows how Microsoft SQL Server™ 2000 assigns pages when it creates a table.

1. Display the topic slide and point out the existence of two mixed extents and two uniform extents.
2. Advance to the next animation, where the Index Allocation Map (IAM) page is created along with one data page. Briefly explain that the IAM page contains a reference to the data page.
3. Advance to the next animation, where seven more data pages are assigned. Briefly explain that the IAM page contains a reference to those seven data pages, as well.
4. Advance to the next animation, where a uniform extent is assigned. Conclude by explaining that from this point forward only uniform extents are assigned and that the IAM page contains a bitmap identifying each extent.

Module Strategy

Use the following strategy to present this module:

- Creating Databases

  Describe the process of creating a database. Explain how to set database options. Review the system stored procedures that display information about database options.

- Creating Filegroups

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

- Managing Databases

  Describe and compare three methods for managing data and log file growth: configuring files to grow automatically, increasing file size manually, and adding secondary files. Explain how to shrink and drop a database.

- Introduction to Data Structures

  Introduce students to pages and extents. Emphasize that most data is managed in extents, and that mixed extents exist to efficiently manage small tables. Review the different types of pages, but keep the discussion general.

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 2073A, *Programming a Microsoft SQL Server 2000 Database*.

Lab Setup

There are no lab setup requirements that affect replication or customization.

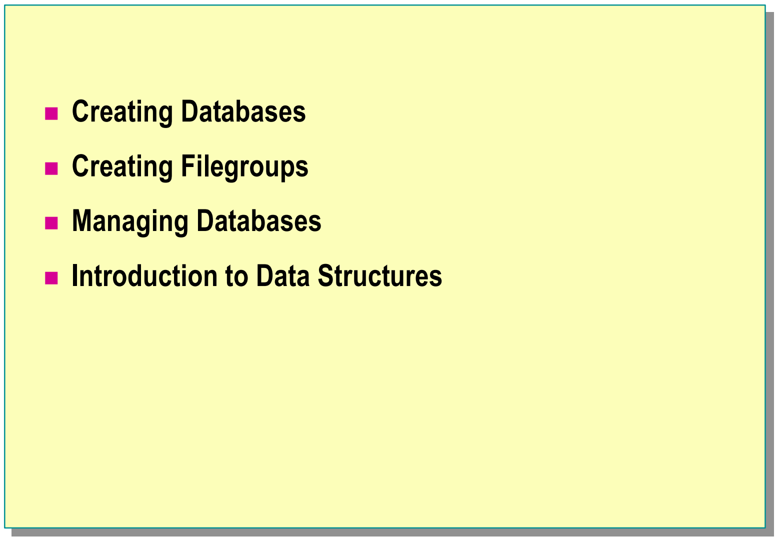
Lab Results

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 and how SQL Server stores data.



- **Creating Databases**
- **Creating Filegroups**
- **Managing Databases**
- **Introduction to Data Structures**

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

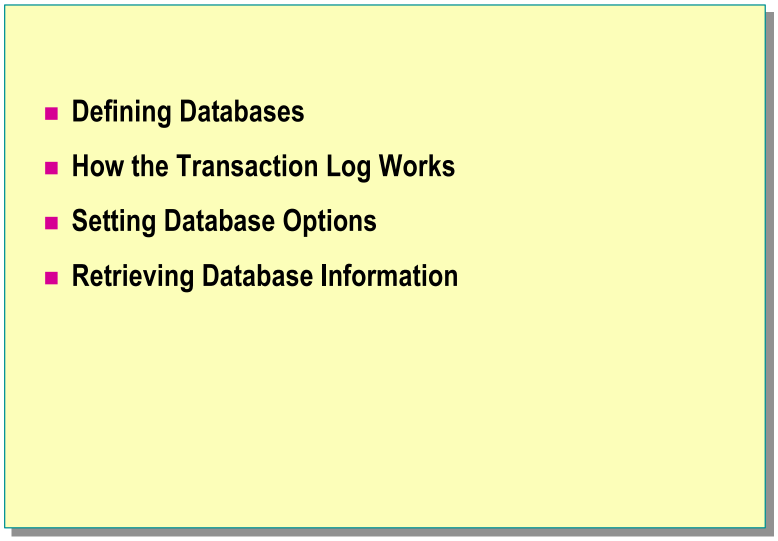
After completing this module, you will be able to:

- Create a database.
- Create a filegroup.
- Manage a database.
- Describe data structures.

◆ Creating Databases

Topic Objective
To describe how to create databases.

Lead-in
This section describes how to create databases.



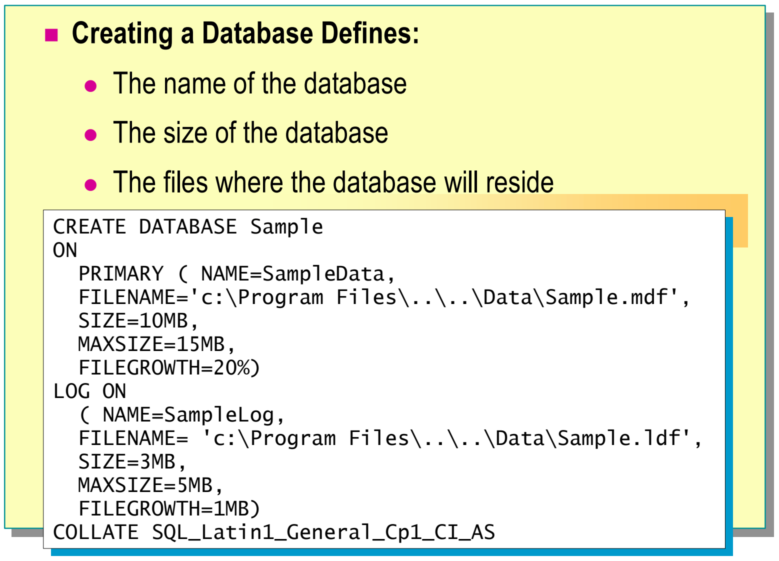
- **Defining Databases**
- **How the Transaction Log Works**
- **Setting Database Options**
- **Retrieving Database Information**

This section describes how to create databases, specify database options, and retrieve database information. It also describes how the transaction log works.

Defining Databases

Topic Objective
To describe how to define a database.

Lead-in
You can define a database by using SQL Server Enterprise Manager, or by executing Transact-SQL statements.



- **Creating a Database Defines:**
  - The name of the database
  - The size of the database
  - The files where the database will reside

```
CREATE DATABASE Sample
ON
  PRIMARY ( NAME=SampleData,
  FILENAME='c:\Program Files\..\..\Data\Sample.mdf',
  SIZE=10MB,
  MAXSIZE=15MB,
  FILEGROWTH=20%)
LOG ON
  ( NAME=SampleLog,
  FILENAME= 'c:\Program Files\..\..\Data\Sample.ldf',
  SIZE=3MB,
  MAXSIZE=5MB,
  FILEGROWTH=1MB)
COLLATE SQL_Latin1_General_Cp1_CI_AS
```

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

You can define a database by using SQL Server Enterprise Manager or the CREATE DATABASE statement in SQL Query Analyzer. The process of defining a database also creates a transaction log for that database.

Information about each database in SQL Server is stored in the **sysdatabases** table in the **master** database. Therefore, you must use the **master** database to define a database when you use Transact-SQL.

Defining a database is a process of specifying the name of the database and designating the size and location of the database files. When the new database is created, it is a duplicate of the **model** database. Any options or settings in the **model** database are copied into the new database.

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

Syntax

```
CREATE DATABASE database_name
  [ON
    { [PRIMARY] (NAME = logical_file_name,
        FILENAME = 'os_file_name'
        [, SIZE = size]
        [, MAXSIZE = {max_size | UNLIMITED}]
        [, FILEGROWTH = growth_increment] )
    } [,...n]
  ]
  [LOG ON
    { (NAME = logical_file_name,
        FILENAME = 'os_file_name'
        [, SIZE = size]
        [, MAXSIZE = {max_size | UNLIMITED}]
        [, FILEGROWTH = growth_increment] )
    } [,...n]
  ]
  [COLLATE collation_name]
```

When you create a database, you can set the following parameters:

PRIMARY This parameter specifies the files in the primary filegroup. The primary filegroup contains all of the database system tables. It also contains all objects not assigned to user filegroups. Every database has one primary data file. The primary data file is the starting point of the database and points to the rest of the files in the database. The recommended file name extension for primary data files is .mdf. If you do not specify the PRIMARY keyword, the first file listed in the statement becomes the primary file.

FILENAME This parameter specifies the operating system file name and path for the file. The path in the *os_file_name* must specify a folder on the server on which SQL Server is installed.

SIZE This parameter specifies the size of the data or log file. You can specify sizes in megabytes (MB)—the default value—or kilobytes (KB). The minimum size is 512 KB for both the data and log file. The size specified for the primary data file must be at least as large as the primary file of the **model** database. When adding a data file or log file, the default value is 1 MB.

MAXSIZE This parameter specifies the maximum size to which the file can grow. You can specify sizes in megabytes—the default value—or kilobytes. If you do not specify a size, the file grows until the disk is full.

You can specify file growth in three ways: in megabytes, in kilobytes, or as a percentage. The percentage only applies to file growth, not maximum size.

FILEGROWTH This parameter specifies the growth increment of the file. The FILEGROWTH setting for a file cannot exceed the MAXSIZE setting. A value of 0 indicates no growth. The value can be specified in megabytes—the default—in kilobytes, or as a percentage (%). The default value if FILEGROWTH is not specified is 10 percent, and the minimum value is 64 KB (one extent). The specified size is rounded to the nearest 64 KB.

COLLATION This parameter specifies the default collation for the database. Collation includes the rules governing the use of characters for either a language or an alphabet.

Example

The following example creates a database called **Sample** with a 10-MB primary data file and a 3-MB log file in a default instance of SQL Server.

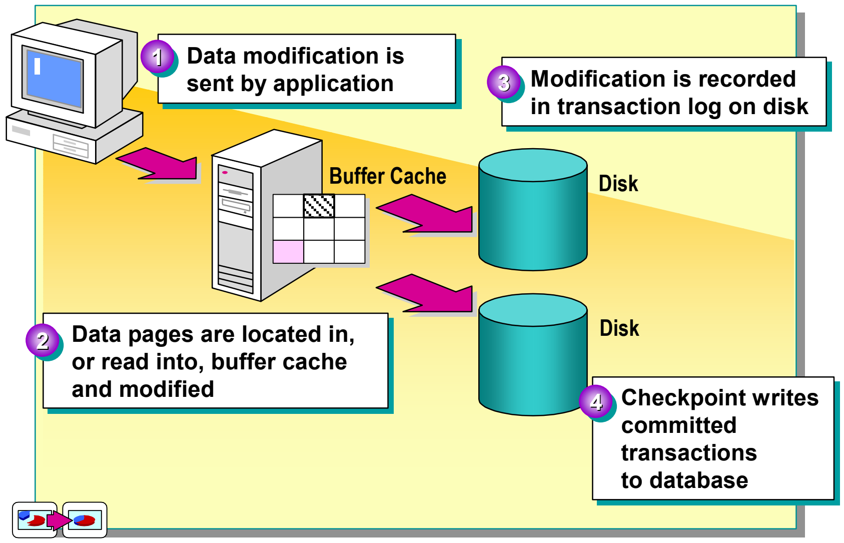
Delivery Tip
Mention that the file path will vary if a named instance of SQL Server is used instead of a default instance.

```
CREATE DATABASE Sample
ON
  PRIMARY ( NAME=SampleData,
  FILENAME='c:\Program Files\
    Microsoft SQL Server\MSSQL\Data\Sample.mdf',
  SIZE=10MB,
  MAXSIZE=15MB,
  FILEGROWTH=20%)
LOG ON
  ( NAME=SampleLog,
  FILENAME='c:\Program Files\
    Microsoft SQL Server\MSSQL\Data\Sample.ldf',
  SIZE=3MB,
  MAXSIZE=5MB,
  FILEGROWTH=1MB)
COLLATE SQL_Latin1_General_Cp1_CI_AS
```

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.


1 Data modification is sent by application
3 Modification is recorded in transaction log on disk
Buffer Cache
Disk
Disk
2 Data pages are located in, or read into, buffer cache and modified
4 Checkpoint writes committed transactions to database

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

SQL Server records every transaction in a transaction log to maintain database consistency and to 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.

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 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.

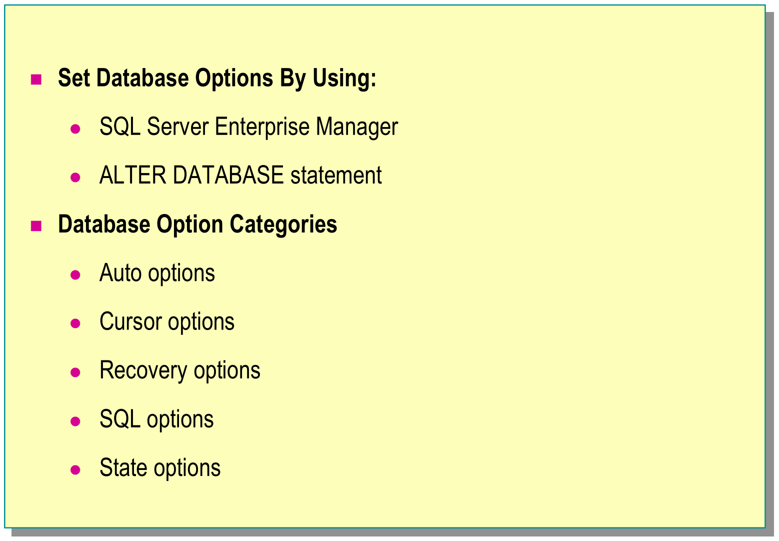
If the system fails, the automatic recovery process uses the transaction log to roll forward all committed transactions and roll back any incomplete transactions.

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.

Setting Database Options

Topic Objective
To list some of the configurable database options.

Lead-in
For most databases, you can configure a number of options.



- **Set Database Options By Using:**
  - SQL Server Enterprise Manager
  - ALTER DATABASE statement
- **Database Option Categories**
  - Auto options
  - Cursor options
  - Recovery options
  - SQL options
  - State options

After you have created a database, you can set the database options by using SQL Server Enterprise Manager or the ALTER DATABASE statement.

You can configure a number of database options, but you are able to set them for only one database at a time. To affect options in all new databases, change the **model** database.

The following table lists some of the more frequently used options.

Database option category	Database option	Description
Auto options	**AUTO_CREATE_STATISTICS**	Automatically creates any missing statistics needed by a query for optimization. The default is ON.
	AUTO_UPDATE_STATISTICS	Automatically updates out-of-date statistics required by a query for optimization. The default is ON.
Cursor options	**CURSOR_CLOSE_ON_COMMIT**	Automatically closes open cursors when a transaction is committed. The default is OFF, and cursors remain open.
	CURSOR_DEFAULT LOCAL | GLOBAL	CURSOR_DEFAULT_LOCAL limits the scope of the cursor. It is local to the batch, stored procedure, or trigger in which the cursor was created. CURSOR_DEFAULT_GLOBAL is the default setting; the scope of the cursor is global to the connection.

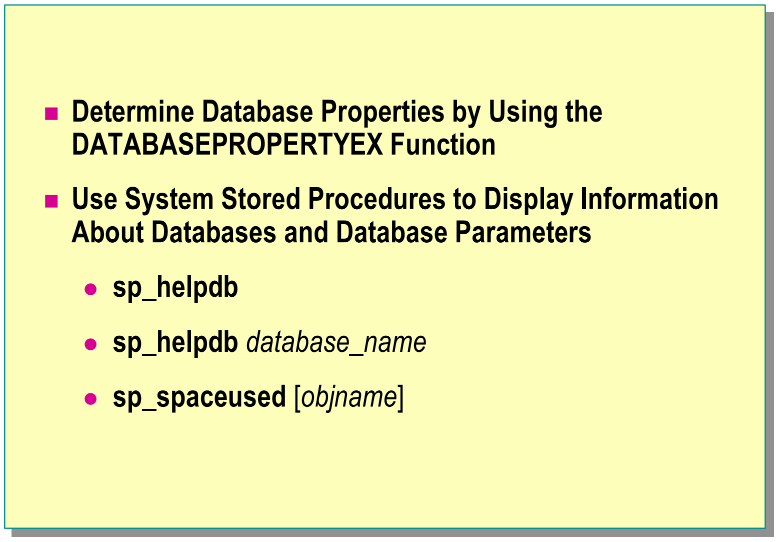
(continued)

Database option category	Database option	Description
Recovery options	**RECOVERY FULL | BULK_LOGGED | SIMPLE**	FULL provides full recoverability from media failure; it is the default. BULK_LOGGED uses less log space because logging is minimal, but it has greater risk of exposure. SIMPLE recovers the database only to the last full database backup or last differential backup.
	TORN_PAGE_DETECTION	Allows SQL Server to detect incomplete I/O operations caused by power failures or other system outages. The default is ON.
SQL options	**ANSI_NULL_DEFAULT**	Allows the user to control the database default nullability. SQL Server 2000 defaults to NOT NULL.
	ANSI_NULLS	When ON, all comparisons to a null value evaluate to NULL (unknown). When OFF, comparisons of non-Unicode values to a null value evaluate to TRUE if both values are NULL. By default, the ANSI_NULLS database option is OFF.
State options	**READ_ONLY | READ_WRITE**	Defines a database as read-only—use to set security for decision-support databases—or returns database to read/write operations.
	SINGLE_USER | RESTRICTED_USER | MULTI_USER	**SINGLE_USER** allows one user at a time to connect to the database. All other user connections are broken. **RESTRICTED_USER** allows only members of the **db_owner** fixed database role and **dbcreator** and **sysadmin** fixed server roles to connect to the database. **MULTI_USER** allows all users with the appropriate permissions to connect to the database. **MULTI_USER** is the default setting.

Retrieving Database Information

Topic Objective
To list the various ways to retrieve database information.

Lead-in
There are several ways to retrieve database information.



- **Determine Database Properties by Using the DATABASEPROPERTYEX Function**
- **Use System Stored Procedures to Display Information About Databases and Database Parameters**
  - **sp_helpdb**
  - **sp_helpdb** *database_name*
  - **sp_spaceused** [*objname*]

You can determine database properties by using the DATABASEPROPERTYEX function.

Syntax

SELECT DATABASEPROPERTYEX (*database*, *property*)

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

The following table lists some of the database properties.

Collation	IsFulltextEnabled
IsAnsiNullDefault	IsInStandBy
IsAnsiNullsEnabled	IsNullConcat
IsAnsiPaddingEnabled	IsQuotedIdentifiersEnabled
IsAnsiWarningsEnabled	IsRecursiveTriggersEnabled
IsArithmeticAbortEnabled	Recovery
IsAutoCreateStatistics	Status
IsAutoShrink	Updateability
IsAutoUpdateStatistics	UserAccess
IsCloseCursorsOnCommitEnabled	Version

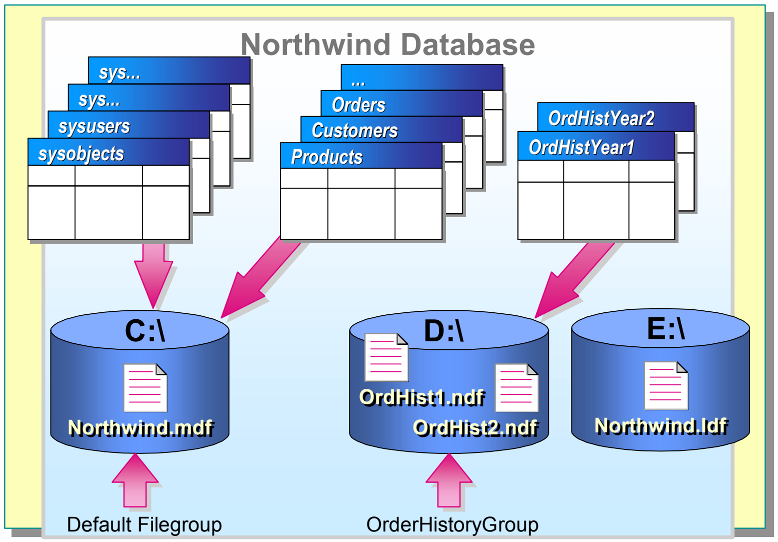
The following table lists commonly-used system stored procedures that display information about databases and database parameters.

System stored procedure	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.

Creating Filegroups

Topic Objective
To introduce the concept of filegroups.

Lead-in
Filegroups are a named collection of one or more files that form a unit of allocation and administration.



If your hardware setup includes multiple disk drives, you can locate specific objects and files on individual disks, grouping your database files into filegroups. *Filegroups* are named collections of files. SQL Server includes one filegroup as a default. You can create additional filegroups by using either the CREATE DATABASE or ALTER DATABASE statement.

With filegroups, you can locate specific objects on a specific file. In the illustration, the OrdHist1.ndf and OrdHist2.ndf files are placed on separate disk to separate files that are heavily queried from those that are heavily modified and to reduce disk drive contention.

System administrators also can back up and restore individual files or filegroups instead of backing up or restoring an entire database. Backing up files or filegroups is necessary on large databases to have an effective back up and restore strategy.

Considerations When Using Filegroups

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

Use of filegroups is an advanced database design technique. You must understand your database structure, data, transactions, and queries thoroughly to determine the best way to place tables and indexes on specific filegroups. In many cases, using the striping capabilities of RAID systems provides much of the same performance gain that you might achieve by using filegroups without the added administrative burden of defining and managing them.

Note Log files are not part of a filegroup. Log space is managed separately from data space.

Types of Filegroups

SQL Server offers the following two types of filegroups:

- The primary filegroup, which contains the system tables in the primary data file.
- User-defined filegroups, which are any filegroups that are specified by using the FILEGROUP keyword.

Designating the Default Filegroup

When you create a database, the primary filegroup automatically becomes the default filegroup. The default filegroup receives all new tables, indexes, and files for which a filegroup is not specified. If your database contains more than one filegroup, it is recommended that you change the default to be one of your user-defined filegroups. This prevents the primary filegroup, which contains the system tables, from being unexpectedly filled by a user table.

Sizing the Primary Default Filegroup

If the default filegroup remains the primary filegroup, sizing this filegroup correctly is important. If the filegroup runs out of space, you are not able to add any new information to the system tables. If a user-defined filegroup runs out of space, only the user files that are specifically allocated to that filegroup are affected.

Example

The following example creates a user-defined filegroup in the **Northwind** database and adds a secondary data file to the user-defined filegroup.

```
ALTER DATABASE Northwind
ADD FILEGROUP OrderHistoryGroup
GO

ALTER DATABASE Northwind
ADD FILE
  ( NAME = 'OrdHistYear1',
  FILENAME = 'c:\ Program Files\
    Microsoft SQL Server\MSSQL\Data\OrdHist1.ndf,
  SIZE = 5MB),
TO FILEGROUP OrderHistoryGroup
GO
```

Viewing Filegroup Information

Information about filegroups is available by using functions, such as FILE_NAME, FILE_ID, FILE_PROPERTY, FILEGROUP_NAME, FILEGROUP_ID, and FILEGROUP_PROPERTY. The system stored procedures in the following table also display information about filegroups.

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.

◆ Managing Databases

Topic Objective
To provide an overview of this topic.

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

- **Managing Data and Log File Growth**
- **Monitoring and Expanding a Transaction Log**
- **Shrinking a Database or File**
- **Dropping a Database**

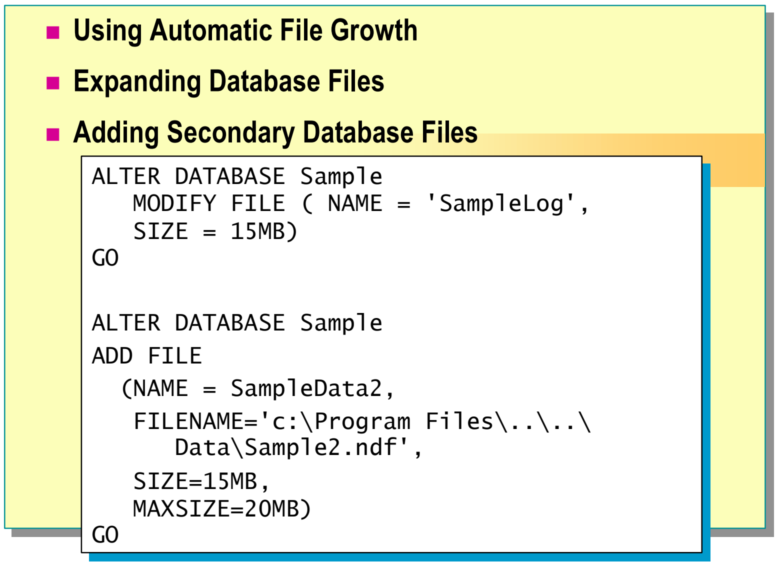
Delivery Tip
Ask students if they are familiar with the topics in this section. If they are, you can either teach this section at a high level, or skip the section entirely.

As your database grows or changes, you can expand or shrink the database size automatically or manually. When you no longer need a database, you can drop it, along with all associated files.

Managing Data and Log File Growth

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

Lead-in
There are several ways to control the size of a database.



- Using Automatic File Growth
- Expanding Database Files
- Adding Secondary Database Files

```
ALTER DATABASE Sample
   MODIFY FILE ( NAME = 'SampleLog',
   SIZE = 15MB)
GO

ALTER DATABASE Sample
ADD FILE
  (NAME = SampleData2,
   FILENAME='c:\Program Files\..\..\
      Data\Sample2.ndf',
   SIZE=15MB,
   MAXSIZE=20MB)
GO
```

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 or decreasing the current or maximum size of existing database and log files.
- Manually adding secondary database and log files.

Using Automatic File Growth

You can set the automatic file growth option by using the ALTER DATABASE statement or SQL Server Enterprise Manager to specify that database files automatically expand by a specified amount whenever necessary. Using automatic file growth reduces the administrative tasks involved with manually increasing the database size.

You can specify the initial size, maximum size, and growth increment of each file. Although it is possible to specify file growth in megabytes or kilobytes, you should specify file growth by percentage. If you do not specify a maximum size, a file can continue to grow until it uses all available space on the disk.

When you use automatic file growth with multiple files, SQL Server uses a proportional fill strategy across all the files in each filegroup. As data is written to the filegroup, SQL Server writes an amount proportional to the free space in the file to each file in the filegroup, instead of writing all the data to the first file until it is full and then writing to the next file.

For optimum performance:

- Allocate sufficient initial size to the database and the log to avoid frequently activating automatic growth.
- Set a maximum size for data files if you have multiple databases.
- Set the data and log file growth increments to sufficient sizes to avoid frequently activating automatic growth.

  For example, if the log grows by 40 MB daily, set the autogrow increment to 50 MB or 100 MB—rather than to 1 MB.

Expanding Database Files

If you do not configure an existing file to grow automatically, you still can increase its size. A value of zero for the growth increment indicates that it does not grow automatically.

Adding Secondary Database Files

You can create secondary database files to expand the size of a database. Use secondary database files to place data files on separate physical disks when you do not use the disk-striping capabilities of RAID systems.

Partial Syntax

ALTER DATABASE *database*
  { ADD FILE < filespec > [,...n] [TO FILEGROUP *filegroup_name*]
  | ADD LOG FILE < filespec > [,...*n*]
  | REMOVE FILE *logical_file_name* [WITH DELETE]
  | ADD FILEGROUP *filegroup_name*
  | REMOVE FILEGROUP *filegroup_name*
  | MODIFY FILE < filespec >
  | MODIFY NAME = *new_dbname*
  | MODIFY FILEGROUP *filegroup_name*
    {*filegroup_property* | NAME = *new_filegroup_name* }
  | SET < optionspec > [,...*n*] [WITH < termination >]
  | COLLATE < *collation_name* >
  }

Example

The following example increases the current log size and adds a secondary data file to the **Sample** database.

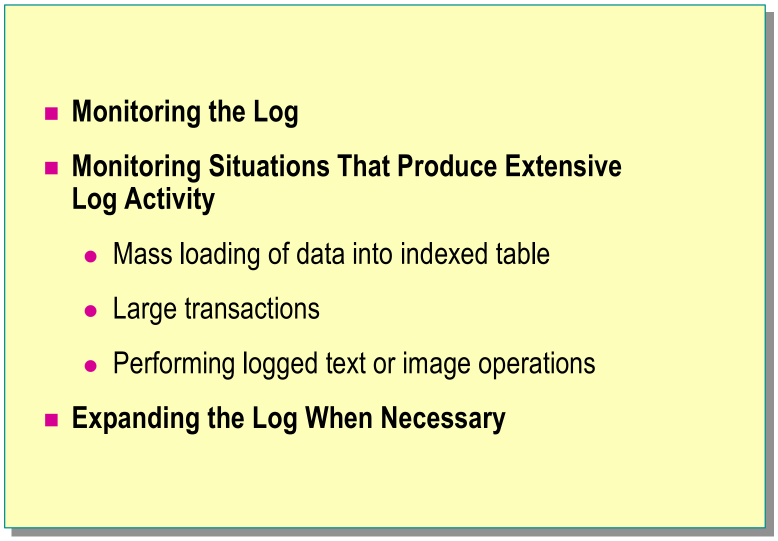
```
ALTER DATABASE Sample
   MODIFY FILE ( NAME = 'SampleLog',
   SIZE = 15MB)
GO

ALTER DATABASE Sample
ADD FILE
(NAME = 'SampleData2' ,
FILENAME='c:\Program Files\
   Microsoft SQL Server\MSSQL\Data\Sample2.ndf',
SIZE=15MB ,
MAXSIZE=20MB)
GO
```

Monitoring and Expanding a Transaction Log

Topic Objective
To show how to monitor and expand transaction log space.

Lead-in
Plan carefully so that you have enough log space.



- **Monitoring the Log**
- **Monitoring Situations That Produce Extensive Log Activity**
  - Mass loading of data into indexed table
  - Large transactions
  - Performing logged text or image operations
- **Expanding the Log When Necessary**

When a database grows, or when data modification activity increases, you may need to expand the transaction log.

Monitoring the Log

Plan carefully so that you do not have too little log space. Monitoring the log on a regular basis helps you determine the optimal time to expand it.

Warning If your transaction log runs out of space, SQL Server cannot record transactions and does not allow changes to your database.

Delivery Tip
Demonstrate how to monitor the log by using System Monitor.

You can monitor the transaction log with SQL Server Enterprise Manager, the DBCC SQLPERF (LOGSPACE) statement, or Microsoft Windows® 2000 System Monitor.

You can monitor the transaction logs of individual databases by using SQL Server:Database object counters in System Monitor. These counters include ones listed in the following table.

Object counter	Displays
Log Bytes Flushed/sec	Number of bytes in the log buffer when buffer is flushed
Log Flushes/sec	Number of log flushes
Log Flush Waits/Sec	Number of commits that are waiting on log flush
Percent Log Used	Percent of space in the log that is in use
Log File(s) Size (KB)	Cumulative size of all of the log files in the database
Log Cache Hit Ratio	Percent of log cache reads that were successful from the log cache

Monitoring Situations That Produce Extensive Log Activity

Some situations that produce additional transaction log activity are:

- Loading information into a table that has indexes. SQL Server logs all inserts and index changes. When loading tables without indexes, SQL Server logs only extent allocations.
- Transactions that perform many modifications (INSERT, UPDATE, and DELETE statements) to a table in a single transaction. This typically occurs when the statement lacks a WHERE clause or when the WHERE clause is too general, causing a large number of records to be affected.
- Adding or modifying text or image data in a table.

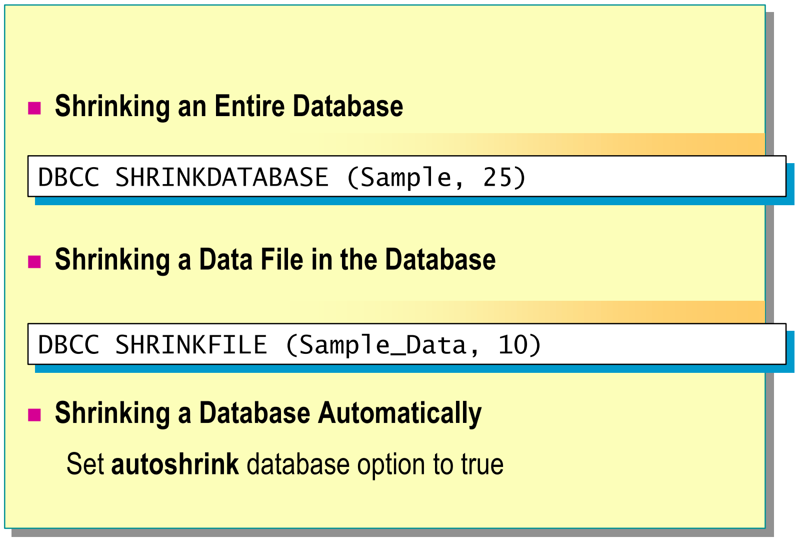
Expanding the Log When Necessary

You can expand the transaction log by using SQL Server Enterprise Manager or the ALTER DATABASE statement.

Shrinking a Database or File

Topic Objective
To show how to shrink a database.

Lead-in
You can shrink an entire database or specific files in a database.



- **Shrinking an Entire Database**

```
DBCC SHRINKDATABASE (Sample, 25)
```

- **Shrinking a Data File in the Database**

```
DBCC SHRINKFILE (Sample_Data, 10)
```

- **Shrinking a Database Automatically**

  Set **autoshrink** database option to true

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

Shrinking an Entire Database

You can shrink an entire database by using SQL Server Enterprise Manager or by executing the Database Consistency Checker (DBCC) statement SHRINKDATABASE. This shrinks the size of all data files in the database.

SQL Server shrinks log files by using a deferred shrink operation, and does so as if all of the log files existed in one contiguous log pool. Log files are reset when the log is truncated; SQL Server attempts to shrink the truncated log files to as close to the targeted size as possible.

Syntax

DBCC SHRINKDATABASE (*database_name* [, *target_percent*] [,
{NOTRUNCATE | TRUNCATEONLY}])

The following table describes the DBCC SHRINKDATABASE options.

Option	Description
target_percent	Specifies the wanted percentage of free space left in the database file after SQL Server has shrunk the database.
NOTRUNCATE	Causes SQL Server to retain the freed file space in the database files. The default is to release the freed file space to the operating system.
TRUNCATEONLY	Causes any unused space in the data files to be released to the operating system and shrinks the file to the last allocated extent, reducing the file size without moving any data. No attempt is made to relocate rows to unallocated pages. SQL Server ignores *target_percent* when you use this option.

Example

This example shrinks the size of the **SampleData** so that the file will have free space of 25 percent.

```
DBCC SHRINKDATABASE (SampleData, 25)
```

In the preceding example, if the **Sample** database file contains 6 MB of data, the new size of the database will be 8 MB (6 MB of data, 2 MB of free space).

Note SQL Server does not shrink a file to a size smaller than the amount of space that the data occupies. Also, it does not shrink a file beyond the size specified in the SIZE parameter of the CREATE DATABASE statement.

Shrinking a Data File in the Database

You can shrink a data file in a database by using SQL Server Enterprise Manager or by executing the DBCC statement SHRINKFILE.

Syntax

DBCC SHRINKFILE ({*file_name* | *file_id*} [, *target_size*] [,
  { EMPTYFILE | NOTRUNCATE | TRUNCATEONLY}])

The following table describes the DBCC SHRINKFILE options.

Option	Description
target_size	Specifies the wanted size for the data file in megabytes, expressed as an integer. If not specified, DBCC SHRINKFILE reduces the size as much as possible.
EMPTYFILE	Migrates all data from the specified file to other files in the same filegroup. SQL Server no longer allows data to be placed on the file used with the EMPTY_FILE option. Use this option to drop the file by using the ALTER DATABASE statement.

Example

This example shrinks the size of the **sample** data file to 10 MB.

```
DBCC SHRINKFILE (Sample, 10)
```

Shrinking a Database Automatically

Autoshrink is not enabled by default. By setting the **autoshrink** database option to true, you can set a database option to recover unused space automatically. You can also change this option with SQL Server Enterprise Manager.

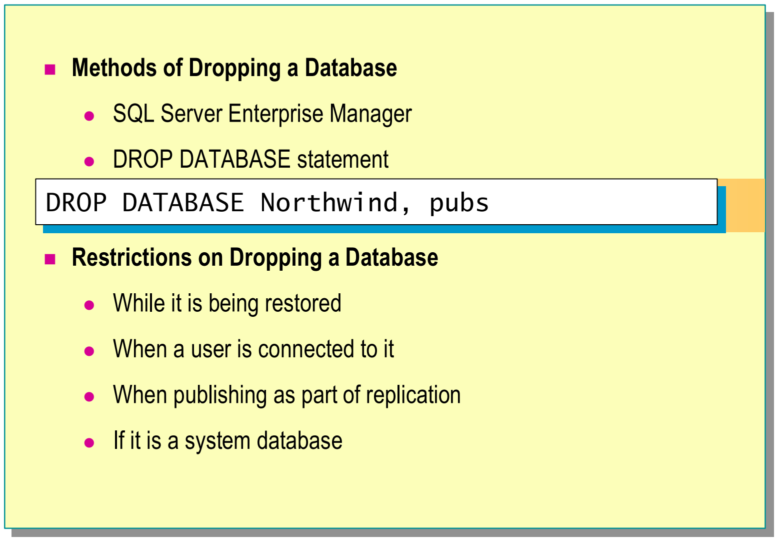
Consider the following facts and guidelines when you shrink a database or a data file:

- The resulting database must be larger than the size of the **model** database or the existing data in the database or data file.
- Before you shrink a database or a data file, you should back up the database, and the **master** database.
- DBCC SHRINKDATABASE and SHRINKFILE perform some actions on a deferred basis, so you may not see the database or file size reduced immediately.
- DBCC SHRINKFILE can reduce the size of a database to smaller than the size specified when the database was created or altered, but not to smaller than the size that the data occupies.

Dropping a Database

Topic Objective
To illustrate how to drop a database.

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



- **Methods of Dropping a Database**
  - SQL Server Enterprise Manager
  - DROP DATABASE statement

```
DROP DATABASE Northwind, pubs
```

- **Restrictions on Dropping a Database**
  - While it is being restored
  - When a user is connected to it
  - When publishing as part of replication
  - If it is a system database

You can drop a database when you no longer need it. Dropping a database deletes the database and the disk files that the database uses.

Methods of Dropping a Database

You can drop databases by using SQL Server Enterprise Manager or by executing the DROP DATABASE statement.

Syntax

DROP DATABASE *database_name* [,…*n*]

Example

This example drops multiple databases by using one statement.

```
DROP DATABASE Northwind, pubs
```

When you drop a database, consider the following facts and guidelines:

- With SQL Server Enterprise Manager, you can drop only one database at a time.
- With Transact-SQL, you can drop several databases at once.
- After you drop 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 drop a database.

Delivery Tip
Mention that SQL Server does not let you drop **master**, **model**, and **tempdb** databases but does allow you to drop the **msdb** system database.

Restrictions on Dropping a Database

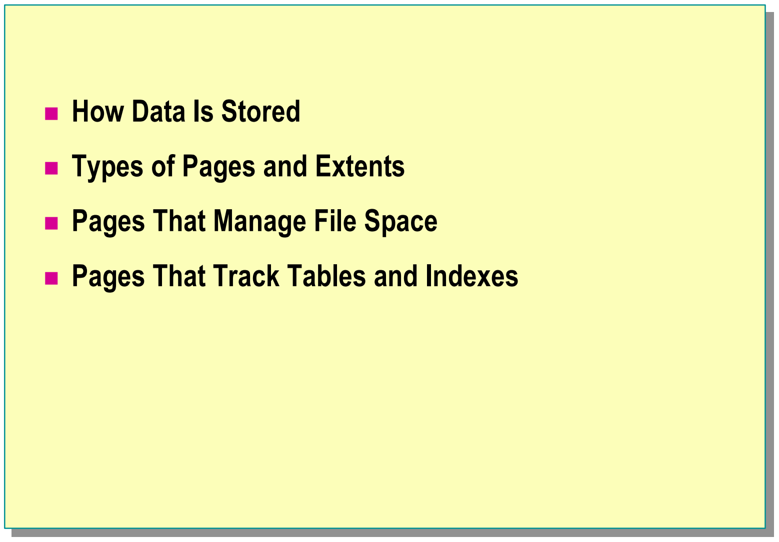
The following restrictions apply to dropping databases. You cannot drop:

- 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.

◆ Introduction to Data Structures

Topic Objective
To introduce data structures.

Lead-in
In this section, you'll learn about the SQL Server data structures.



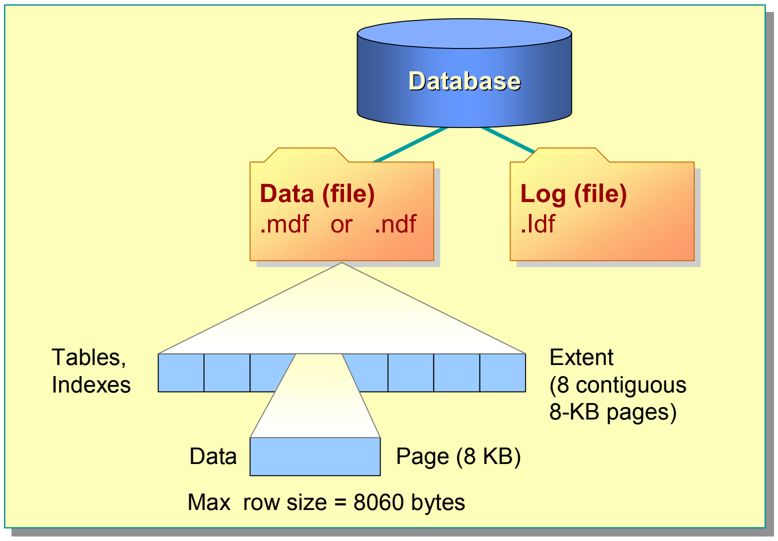
- **How Data Is Stored**
- **Types of Pages and Extents**
- **Pages That Manage File Space**
- **Pages That Track Tables and Indexes**

This section describes the data structures that SQL Server uses to store data.

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.



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 database. Consider the following facts and guidelines about data storage:

Delivery Tip
It is important to know the maximum size of rows and extents when designing a database or when capacity planning.

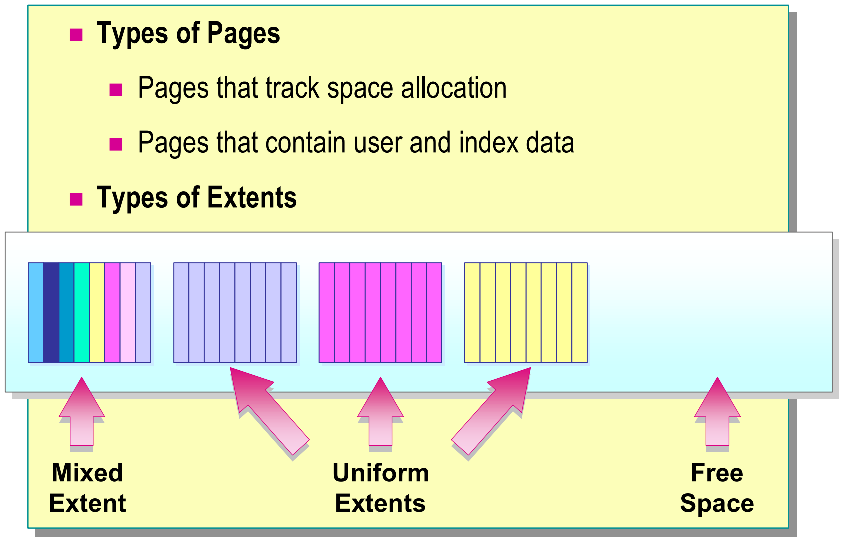
For example, because rows cannot span pages, only one row of 4035 bytes can fit on an 8060-byte page.

- All databases have a primary data file, identified by the .mdf file name extension, and one or more transaction log files, identified by the .ldf file name extension. A database also may have secondary data files, which are identified by the .ndf file name extension. These physical files have both operating system file names and logical file names that you can use in Transact-SQL statements.
- When you create a database, a copy of the **model** database, which includes the system tables, is copied to the database. The minimum size of a database must be equal to or greater than the size of the **model** database.
- SQL Server stores, reads, and writes data in 8-KB blocks of contiguous disk space called *pages*. This means that a database can store 128 pages per megabyte.
- Rows cannot span pages. Thus, the maximum amount of data in a single row, subtracting the space required for row overhead, is 8060 bytes.
- All pages are stored in *extents*. An extent is eight contiguous pages, or 64 KB. Therefore, a database has 16 extents per megabyte.
- Transaction log files hold all of the information necessary for recovery of the database in the event of a system failure. By default, the size of the transaction log is 25 percent of the size of the data files. Use this figure as a starting point and adjust it according to the needs of your application.

Types of Pages and Extents

Topic Objective
To introduce the types of pages and extents.

Lead-in
Most allocation in the database occurs in eight-page blocks called extents.



- **Types of Pages**
  - Pages that track space allocation
  - Pages that contain user and index data
- **Types of Extents**

Pages and extents are the primary data structures in the SQL Server physical database.

Types of Pages

SQL Server uses several types of pages: some track space allocation, and some contain user and index data. The pages that track allocation contain densely packed information. This allows SQL Server to efficiently keep them in memory for easy tracking.

Types of Extents

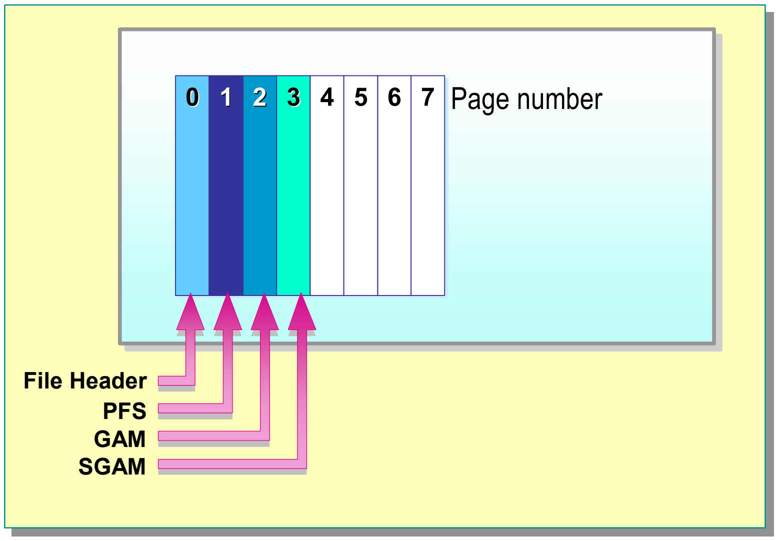
SQL Server uses two types of extents:

- Extents that contain pages from two or more objects are called *mixed extents*. Every table starts as a mixed extent. You use mixed extents primarily for pages that track space and contain small objects.
- Extents that have all eight pages allocated to a single object are called *uniform extents*. They are used when tables or indexes need more than 64 KB of space.

Pages That Manage File Space

Topic Objective
To describe the pages that manage file space.

Lead-in
The first four pages of each file tell SQL Server which pages are in use.



Delivery Tip
Emphasize the following points: (1) The PFS tracks each page. (2) The GAM and SGAM track extents, not pages. (3) By using both the GAM and SGAM SQL Server can find space when new pages are needed. (4) PFS, GAM, and SGAM pages are repeated as needed in large files.

The first extent of each file is a mixed extent that contains a file header page followed by three allocation pages. SQL Server allocates this mixed extent when you create the primary data file and uses these pages internally.

File Header Page

The file header page contains the attributes of the file, such as the name of the database that owns the file, its file group, minimum size, and growth increment. It is the first page in every file (Page 0).

PFS Page

The Page Free Space (PFS) page is an allocation page that contains information about free space available on the pages in a file. Page 1 of each file is a PFS page. SQL Server adds other PFS pages as needed.

Each PFS page can track 8,000 contiguous pages, which is nearly 64 MB of data. For each page, the PFS page contains a byte that tracks:

- Whether the page has been allocated.
- Whether the page is on a mixed or uniform extent.
- An approximation of how much space is available on the page.

GAM and SGAM Pages

SQL Server uses Global Allocation Map (GAM) and Secondary Global Allocation Map (SGAM) pages to determine the location of free extents or mixed extents with free pages.

GAM Pages

The GAM page is an allocation page that contains information about allocated extents. Page 2 of each file is a GAM page. SQL Server adds additional GAM pages as needed.

Each GAM page covers 63,904 extents, or nearly 4 gigabytes (GB) of data. The GAM page contains one bit for each extent that it covers. The bit is set to 0 if the extent is allocated, and set to 1 if it is free.

SGAM Pages

The SGAM page is an allocation page that contains information about allocated mixed extents. Page 3 of each file is an SGAM page. SQL Server adds additional SGAM pages as needed.

SGAM pages track mixed extents that currently have at least one unused page. They also cover 63,904 extents. A bit set to 0 indicates that an extent is either a uniform extent or a mixed extent without any free pages. A bit set to 1 indicates a mixed extent with one or more free pages.

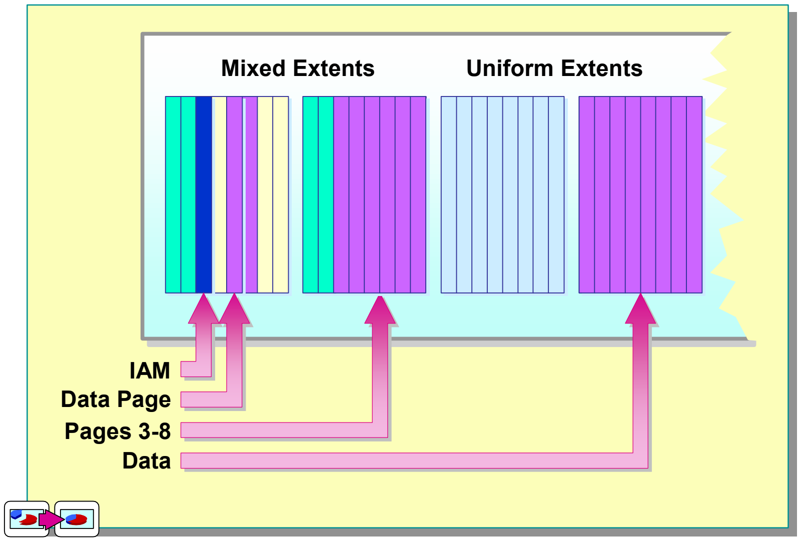
The following table summarizes the setting of the GAM and SGAM bits:

If the GAM bit is set to…	And the SGAM is set to…	Then …
1	0	It is an available extent. This extent is not in use.
0	1	It is an available page. This mixed extent has an unassigned page or pages.
0	0	It is an extent with nothing available. The extent is assigned as a uniform extent or a full mixed extent.

Pages That Track Tables and Indexes

Topic Objective
To examine the role of IAM pages.

Lead-in
IAM pages track the relationship between an object and the extents or pages that it uses.



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

Emphasize that every table and index has its own IAM page. SQL Server can use the IAM page to find every extent and page assigned to the object.

SQL Server initially assigns every table and index an allocation page and at least one data page in a mixed extent. As the object grows, SQL Server assigns up to seven more pages from mixed extents as needed. When the object exceeds eight pages, SQL Server assigns additional pages from uniform extents.

SQL Server uses four types of pages to manage tables and indexes. They may appear at any location in the file. They are the IAM, Data, Text/Image, and Index pages.

IAM Page

The IAM page is an allocation page that contains information about the extents that a table or index uses.

The IAM page contains the location of the eight initial pages and a bitmap of extents indicating which extents are in use for that object. A single IAM page can track up to 512,000 data pages. SQL Server adds more IAM pages for larger tables.

IAM pages are always allocated from mixed extents and can appear anywhere in a file or filegroup. SQL Server attempts to group IAM pages together for speedy retrieval.

Data Page

The Data page contains content other than **text**, **ntext**, and **image** data.

Text/Image Page

The Text/Image page contains **text**, **ntext**, and **image** content.

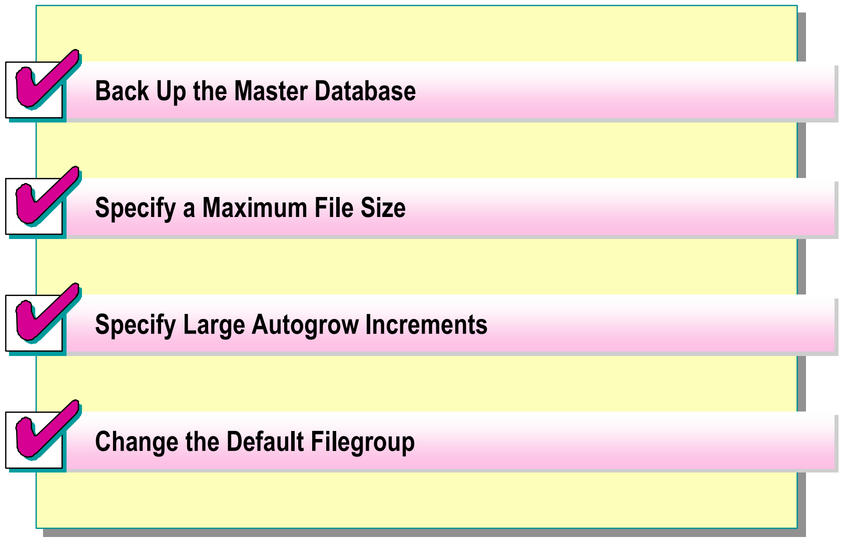
Index Page

The Index page contains index structures.

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.



- Back Up the Master Database
- Specify a Maximum File Size
- Specify Large Autogrow Increments
- Change the Default Filegroup

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

- Back up the **master** database immediately after you create or modify a database.

  This is important because the **master** database has the **system** catalog.
- Specify a maximum size when you use automatic file growth.

  This will prevent any single file from filling the entire hard disk.
- Specify large autogrow increments to avoid frequent file growth.

  This will reduce SQL Server administrative activity and help keep the file from becoming fragmented on the hard disk.
- Change the default filegroup.

  If your database has multiple filegroups, assign one of the user-defined filegroups as the default. This will prevent any unexpected table growth from adversely affecting the system tables in the primary filegroup.

Lab A: Creating and Managing Databases

Topic Objective
To introduce the lab.

Lead-in
In this lab, you will create and manage the **ClassNorthwind** database and clear the transaction log.

Explain the lab objectives.

Objectives

After completing this lab, you will be able to:

- Create a database.
- Manage the growth of a database.
- Change database options to control how often the transaction log is cleared.

Prerequisites

Before working on this lab, you must have:

- Script files for this lab, which are located in C:\Moc\2073A\Labfiles\L03.
- Answer files for this lab, which are located in C:\Moc\2073A\Labfiles\L03\Answers.

Lab Setup

To complete this lab, you must have completed the prior lab.

Note This course is based on the **Northwind** database. The schema for the **Northwind** database is in Appendix A. The labs use a parallel version of the **Northwind** database that is called **ClassNorthwind**.

For More Information

If you require help when executing files, search SQL Query Analyzer Help for "Execute a query".

Other resources that you can use include:

- The **Northwind** database schema.
- Microsoft 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
Creating the ClassNorthwind Database

In this exercise, you will create the **ClassNorthwind** database and define the files used for data and the transaction log.

► **To create the ClassNorthwind database**

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

A complete Transact-SQL script for this procedure is located in C:\MOC\2073A\Labfiles\L03\Answers\Creabase.sql.

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. Open SQL Server Enterprise Manager.

   You have permission to log in to and administer the installation of SQL Server on your computer because your **SQLAdmin*x*** account is a member of the Microsoft Windows® 2000 local group **Administrators**, which is automatically mapped to the SQL Server **sysadmin** role.

3. Expand Microsoft SQL Servers, expand SQL Server Group, and then expand your computer.
4. Right-click **Databases**, and then click **New Database**.
5. Use the values in the following table to create the **ClassNorthwind** database.

For this parameter	Use this value
Database name	**ClassNorthwind**
Database file name	**ClassNorthwind_Data**
Location	(Default)
Initial size	25 MB
Filegroup	Primary
File growth	10 percent
Maximum file size	100 MB
Transaction log file name	**ClassNorthwind_Log**
Location	(Default)
Initial size	15 MB
File growth	10 percent
Log file maximum size	40 MB

6. After you create the **ClassNorthwind** database, in the console tree, expand **Databases**, and then click **ClassNorthwind**.

   Review the information available in the details pane on the **General** tab.

7. Open SQL Query Analyzer and connect by using Windows 2000 Authentication.

8. Execute the **sp_helpdb** system stored procedure to view information on the **ClassNorthwind** database.

```
EXEC sp_helpdb ClassNorthwind
```

Exercise 2
Managing the Growth of the ClassNorthwind Transaction Log File

In this exercise, you will modify the maximum size of the **ClassNorthwind** transaction log file.

► To increase the size of the ClassNorthwind transaction log file

In this procedure, you will write and execute a statement to increase the maximum size of the **ClassNorthwind** transaction log file to 50 MB and the current log size to 20 MB.

A complete Transact-SQL script for this procedure is located in C:\MOC\2073A\Labfiles\L03\Answers\Altebase.sql.

1. Write and execute a statement that increases the maximum size of the **ClassNorthwind** transaction log file to 50 MB.

```
USE master
GO

ALTER DATABASE ClassNorthwind
 MODIFY FILE (NAME=ClassNorthwind_Log,
 MAXSIZE=50MB)
GO
```

2. Write and execute a statement that increases the current size of the **ClassNorthwind** transaction log file to 25 MB.

```
USE master
GO

ALTER DATABASE ClassNorthwind
 MODIFY FILE (NAME=ClassNorthwind_Log,
 SIZE=25MB)
GO
```

3. Write and execute a statement that increases the growth increment of the **ClassNorthwind** transaction log file to 20 percent.

```
USE master
GO

ALTER DATABASE ClassNorthwind
 MODIFY FILE (NAME=ClassNorthwind_Log,
 FILEGROWTH=20%)
GO
```

4. Execute the **sp_helpdb** system stored procedure to view information on the **ClassNorthwind** database and to verify the changes.

```
EXEC sp_helpdb ClassNorthwind
```

Exercise 3
Setting the Database Recovery Model

In this exercise, you will set the database recovery model to SIMPLE. This will allow SQL Server to reclaim log space after the log space is no longer needed for recovery. It also reduces space requirements.

► **To set the database recovery model**

In this procedure, you will write and execute a statement to set the **ClassNorthwind** database recovery model to SIMPLE. You will use the ALTER DATABASE statement.

A complete Transact-SQL script for this procedure is located in C:\MOC\2073A\Labfiles\L03\Answers\RecovModel.sql.

1. Execute the following statement to turn on the option that clears the transaction log automatically each time that SQL Server performs a checkpoint:

   ```
   ALTER DATABASE ClassNorthwind SET RECOVERY SIMPLE
   GO
   ```

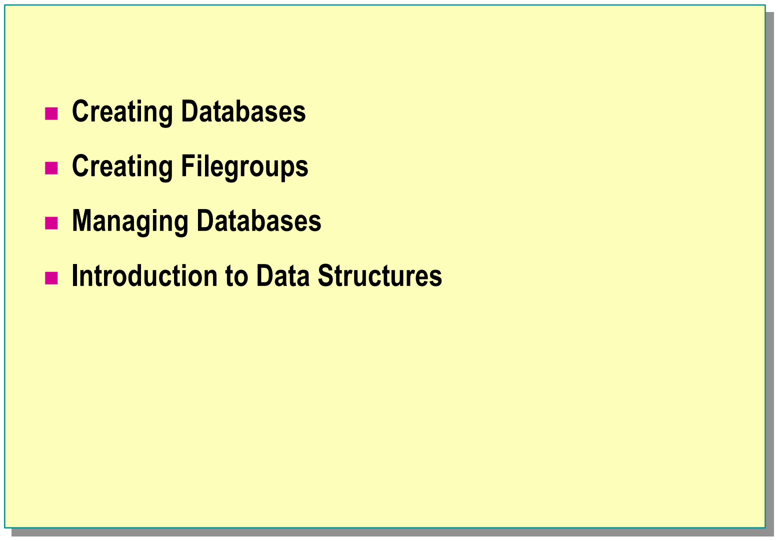
2. Execute the **sp_helpdb** system stored procedure to view information on the **ClassNorthwind** database to verify that the recovery model has been changed.

   ```
   EXEC sp_helpdb ClassNorthwind
   GO
   ```

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.



- **Creating Databases**
- **Creating Filegroups**
- **Managing Databases**
- **Introduction to Data Structures**

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

   Answer varies. It can be in the range of 10–20 percent. You would not want more than 20 percent. Because the database has so little modification activity, it would make sense to allocate closer to 10 percent.

2. What are the advantages of using filegroups?

   You can place tables on specific disks. You can back up large tables separately.

3. You are responsible for managing the mission-critical accounting records of your organization. Which database recovery model would be appropriate for this database?

   You should use the Full Recovery model.

4. The GAM, SGAM, and IAM pages all track data allocation. How is the IAM page different from the GAM and SGAM pages?

   The GAM and SGAM pages track all objects. The IAM page tracks allocation for only one specific table or index.