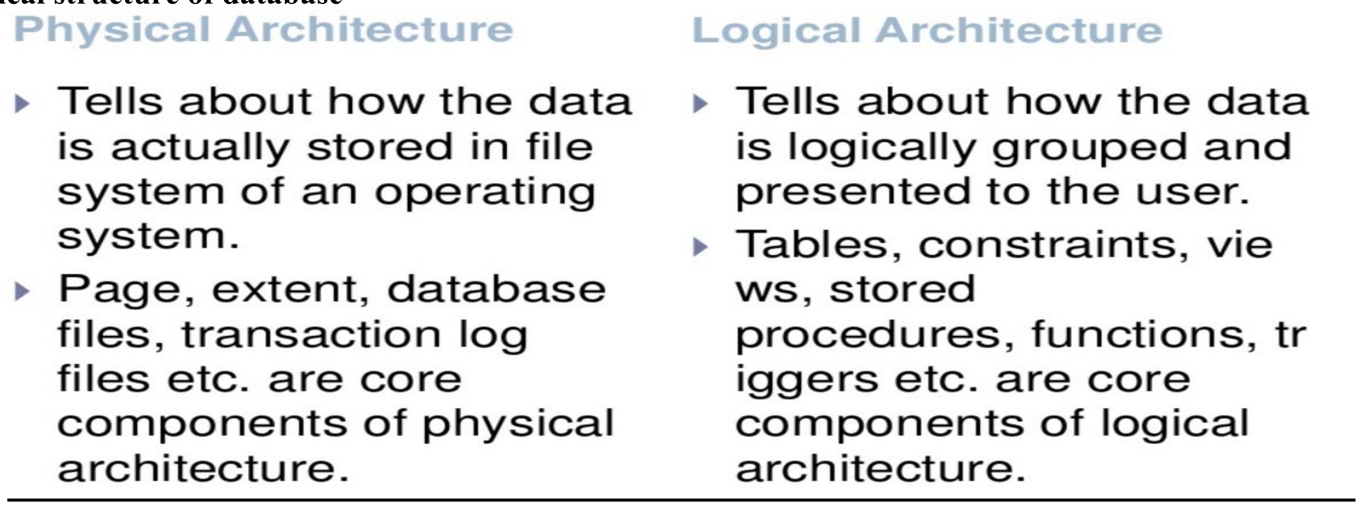
Debere Berhan Polytechnic College

MODULE TITLE: <u>Completing Database Backup and Recovery</u>

LO1: Review database architecture

1.1 Identifying architecture of a database file system

Physical structure of database



Physical Architecture

- Tells about how the data is actually stored in file system of an operating system.
- Page, extent, database files, transaction log files etc. are core components of physical architecture.

Logical Architecture

- Tells about how the data is logically grouped and presented to the user.
- Tables, constraints, vie ws, stored procedures, functions, tr iggers etc. are core components of logical architecture.

- Components of database Architecture include:
  - Page
  - Extent
  - Table
  - Index
  - Database File
  - Database File Group
  - Transaction Log File

i. **Page** is the smallest unit of storage in SQL server.

ii. An **extent** consists of 8 adjacent pages.

iii. An **index** is a set of one or more keys which is used to speed up access to data in a table. It is a separate physical data structure that enables queries to access one or more data rows fast.

iv. **Database File**: SQL Server uses three types of files to map a database to file system of operating system:

1. Primary Data File– It is the starting point of the database and points to the other files in the database. Every database has one primary data file.
2. Secondary Data File is any data file other than primary data file. A database may or may not have secondary data files.
3. Log file - Log files hold all the log information that is used to recover the database. There must be at least one log file for each database.

V. **Database File Group**: Database files can be grouped into file groups for allocation and administration purposes.

File groups can be two types:

- Primary – It contains primary data files and pages of system tables.
- User-defined – These are created by database users.

vi. **Transaction Log File**: DB engine uses transaction logs to maintain integrity of database and for data recovery.

Transaction log file consists of log records of operations performed and are stored sequentially.

Following types of operations are logged

a. The start and end of each transaction. Every data modification (insert, update, or delete) by system stored procedures or data definition language (DDL) statements to any table, including system tables.
b. The logical operation performed: The before and after images of the modified data

1.2. Identifying risks and failure scenarios

Database security concerns the use of a broad range of information security controls to protect databases against compromises of their confidentiality (secrecy or privacy), integrity/reliability and availability (ease of use).
It involves various types or categories of controls, such as technical, procedural/administrative and physical control.

Security risks to database systems include:

- Unauthorized or unintended activity or misuse(use wrongly) by authorized database users, database administrators, or network/systems managers, or by unauthorized users or hackers

  Example: Inappropriate access to sensitive data, metadata or functions within databases, or inappropriate changes to the database programs, structures or security configurations.

- Malware infections causing incidents such as unauthorized access, damage to the data or programs.
- Physical damage to database servers caused by computer room fires or floods, overheating, lightning, electronic breakdowns/equipment failures.
- Data corruption and/or loss caused by the entry of invalid data or commands, mistakes in database or system administration processes, sabotage/criminal damage etc.

Types of information security control appropriate to databases include:

* Access control
* Auditing
* Authentication
* Encryption
* Integrity controls
* Backups

Database Failures can be due to:

- System crashes, resulting in loss of main memory.
- Application software errors.
- Natural physical disasters.
- Carelessness or unintentional destruction of data or facilities.
- Statement Failure: e.g. - When a program attempts to enter invalid data into a database table.
- User Process Failure: e.g. - A user process may be terminated suddenly or unexpectedly
- Network Failure: e.g. - The network interface card or the network connection has failed.
- Instance Failure
  - Hardware failure.
  - A power failure.
  - An emergency shutdown procedure.
- User Error
  - Accidentally dropping a table.
  - Wrongly modify or delete data from a table.
- Media Failures
  - Lose a disk or a disk controller fails or a head crash
  - File corruption
  - The overwriting or deletion of a data file

Lo2: Determine Backup methods appropriate to DB requirement

2.1 Backup and recovery concepts

Database backup is the process of dumping data (from a database, a transaction log, or a file) into backup devices that system creates and maintains.
Recovery is the process of using the backup media to replace uncommitted, inconsistent, or lost data.

Therefore**, Backup and recovery** refers to the various strategies and procedures involved in protecting your database against data loss and reconstructing the database after any kind of data loss.
The SQL Server backup and restore component provides an essential safeguard for protecting critical data stored in your SQL Server databases. To minimize the risk of catastrophic data loss, you need to back up your databases to preserve modifications to your data on a regular basis.

A well-planned backup and restore strategy helps protect databases against data loss caused by a variety of failures.

2.2. Database Recovery techniques

Preventing data loss is one of the most critical issues involved in managing database systems.
Data can be lost as a result of many different problems:

- Disk failures (physical problems)
- Viruses
- Incorrect use of UPDATE and DELETE statements
- Software bug(Program errors) or transaction (some operation) error
- Administrator (human) errors
- Computer failures (system crash) such as **hardware, software or network error**
- Catastrophes (Disasters such as fire, flood, earthquake) or theft

If database has been damaged:
- Need to restore last backup copy of database and reapply updates of committed transactions using log file.

If database is only inconsistent:
- Need to undo changes that caused inconsistency.

To prevent data loss, you can implement a recovery strategy for your databases.

Using Full Database Backups

A full database backup contains all data and database Meta information needed to restore the whole database.
When you restore a full database backup, it restores all database files yielding data in a consistent state from the time the backup completed.
From full database backup, it is possible to recover a database to the state it had when the last backup occurred.

Using Differential Backups

The differential backup stores only the data changes that have occurred since the last full database backup. When the same data has changed many times since the last full database backup, a differential backup stores the most recent version of the changed data. Because it contains all changes since the last full backup, to restore a differential backup, you first need to restore the last full database backup and then apply only the last differential backup.

Using Transaction Log Backups

The transaction log backup is a backup of transaction log entries and contains all transactions that have happened to the database. The main advantages of transaction log backups are as follows:

- Transaction log backups allow you to recover the database to a specific point in time.
- Because transaction log backups are backups of log entries, it is even possible to perform a backup from a transaction log if the data files are destroyed. With this backup, it is possible to recover the database up to the last transaction that took place before the failure occurred.

2.3. Recover immediate update

Immediate update, or UNDO/REDO, is another algorithm to support ABORT and machine failure scenarios.
Database may be **updated** by some operations of a transaction **before** the transaction reaches its **commit point**.

- **Commit** means Make changes done in transaction permanently. A **transaction** is a set of SQL statements.

If a transaction fail after recording some change to the database, but before commit point, **the effect of** its operations **on** the **database must be undone** (transaction must be rollback).

- Undo Operations need to be performed for all unfinished transactions.

2.4. Shadow paging

Shadow paging is a technique for providing atomicity and durability in database system. A **page,** in this context, refers to a unit of physical storage.
Shadow paging is a copy-on-write technique for avoiding in-place updates of pages.

In **shadow paging**, when a write_item operation is performed:

- A new copy of the modified DB page is created and the old copy is not overwritten.
- The new page is written elsewhere on some unused disk block.
- The current directory entry is modified to point to the new disk block.
- The shadow directory is not modified.

Shadow copies; the current copies of data items

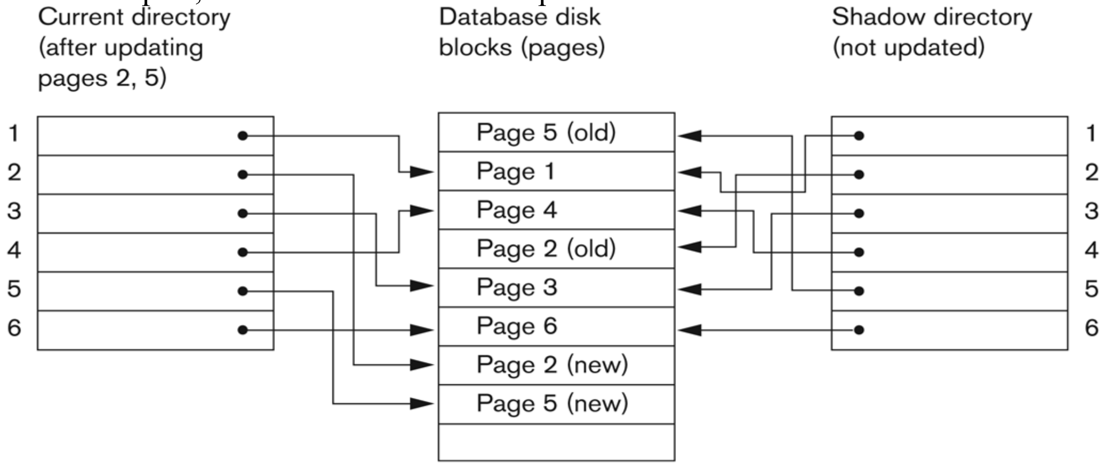

Current directory (after updating pages 2, 5)
Database disk blocks (pages)
Shadow directory (not updated)
1
2
3
4
5
6
Page 5 (old)
Page 1
Page 4
Page 2 (old)
Page 3
Page 6
Page 2 (new)
Page 5 (new)
1
2
3
4
5
6

Figure 19.5
An example of shadow paging.

2.5. The ARIES Recovery Algorithm

Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) is an example of recovery algorithm which is widely used in the database systems. It uses steal/no-force approach for writing the modified buffers back to the database on the disk. This implies that ARIES follows UNDO/REDO technique.

The ARIES recovery algorithm is based on three main principles which are given here.

- Write-ahead logging: This principle states that before making any changes to the database, it is necessary to force-write the log records to the stable storage.
- Repeating history during redo: When the system restarts after a crash, ARIES retraces all the actions of database system prior to the crash to bring the database to the state which existed at the time of the crash. It then undoes the actions of all the transactions that were not committed at the time of the crash.
- Logging changes during undo: A separate log is maintained while undoing a transaction to make sure that the undo operation once completed is not repeated in case the failure occurs during the recovery itself, which causes restart of the recovery process.

The ARIES recovery procedure consists of three main steps:
- Analysis: It identifies the dirty (updated) pages in the buffer and the set of transactions active at the time of crash.

- REDO phase: It actually reapplies updates from the log to the database. Generally the REDO operation is applied to only committed transactions.

- UNDO phase: The log is scanned backwards and the operations of transactions that were active at the time of the crash are undone in reverse order. The information needed for ARIES to accomplish its recovery procedure includes the log, the transaction table, and the dirty page table.

2.6. Recovery in multi DB system

A multi-database consists of a collection of autonomous local databases.
To maintain the atomicity of a multidatabase transaction, it is necessary to have a two-level recovery mechanism.

A global recovery manager or coordinator is needed to maintain information needed for recovery, in addition to the local recovery managers and the information they maintain (log, tables).

In MDBSs, recovering multi-database consistency has a twofold meaning. First, for global transaction aborts, recovering multi-database consistency means to undo the effects of locally committed subsequences belonging to the aborted global transactions from a semantic point of view.

In addition, the effects of transactions which have accessed objects updated by aborted global transactions should be preserved (recall that, after the last operation of a subsequence, all locks held by the subsequence are released).
For the other types of failures, recovering multi-database consistency means to restore the most recent global transaction-consistent state.

2.7. <u>DB Backup from Catastrophic failures</u>

Failure can happen for any number of reasons.

- **User error** is the number one reason for data damage, loss, or corruption.
  This type of failures include: application modifying or destroying the data on its own or through a user choice.
- **Media failure** can happen when the media of the data files or transaction logs failed.
- **Disastrous or catastrophic** event. This can be in the form of fire, flood, or any naturally occurring storm.
  It can also happen through electrical outage, a virus, or the deliberate hacking of your data. Any of these can corrupt or cause the loss of your data.

The recovery manager of a DBMS must be equipped to handle more catastrophic failures such as disk crashes.
The main technique used to handle such crashes is that of database backup. The whole database and the log are periodically copied onto a cheap storage medium such as magnetic tapes. In case of a catastrophic system failure, the latest backup copy can be reloaded from the tape to the disk, and the system can be restarted.
To avoid losing all the effects of transactions that have been executed since the last backup, it is customary to back up the system log at more frequent intervals than full database backup by periodically copying it to magnetic tape.
The system log is usually substantially smaller than the database itself and hence can be backed up more frequently.
Hence, to recover from disk failure, the database is first recreated on disk from its latest backup copy on tape.

LO3. <u>Establish recovery points and disaster recovery procedures</u>

3.1. <u>Determine recovery points</u>

Restoring is the process of copying data from backup and applying logged transactions to the data to roll it forward to the target recovery point. Each backup contain sufficient log to roll back uncommitted transactions to bring the database into a state that is transitionally consistent and usable.
The process of rolling forward uncommitted transactions, if any, and bringing the database online is known as recovery. The goal of roll forward is to return the data to its original state at the recovery point.
The recovery point is the point to which the user specifies that the set of data be recovered.

3.2. <u>Test the restore process</u>

To restore a file from backup device to a location on the hard drive, you have to follow Database Restore and Recovery Procedure.
The basic procedure for performing restore and recovery with RMAN(Recovery Manager) is as follows:

- Determine which database files must be restored from backup, and which backups (which specific tapes, or specific backup sets or image copies on disk) you will use for the restore operation.
- Place the database in the state appropriate for the type of recovery that you are performing.
- Restore lost database files from backup with the `RESTORE` command. You may restore files to their original locations, or you may have to restore them to other locations if, for instance, a disk has failed.
- Perform media recovery on restored data files, if any, with the `RECOVER` command.
- Perform any final steps required to make the database available for users again.

The backup and restore process of a large database or collection of databases on sql server is very important for disaster & recovery purposes.

➢ Backing up and restoration of your MS SQL Server 2008 Database using the Microsoft SQL Server Management Studio.

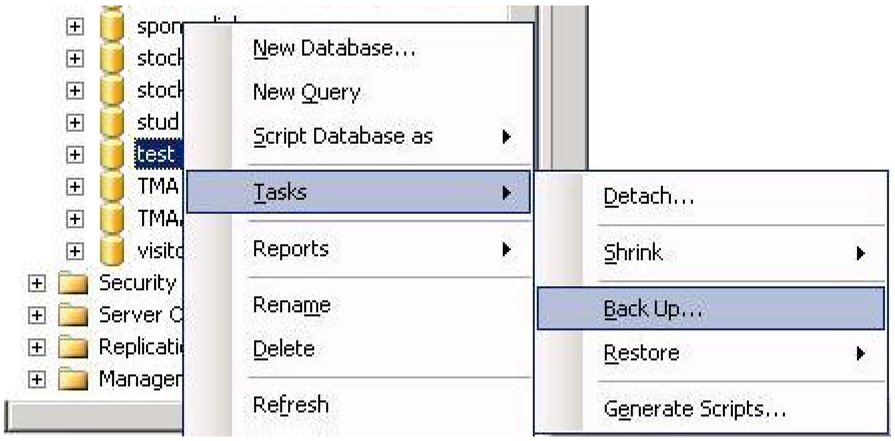
To backup your MS-SQL Server 2008 Database through M**S-SQL Server Management Studio Express,** follow the steps shown below:

First, you need to configure the Microsoft SQL Server Management Studio on your local machine.

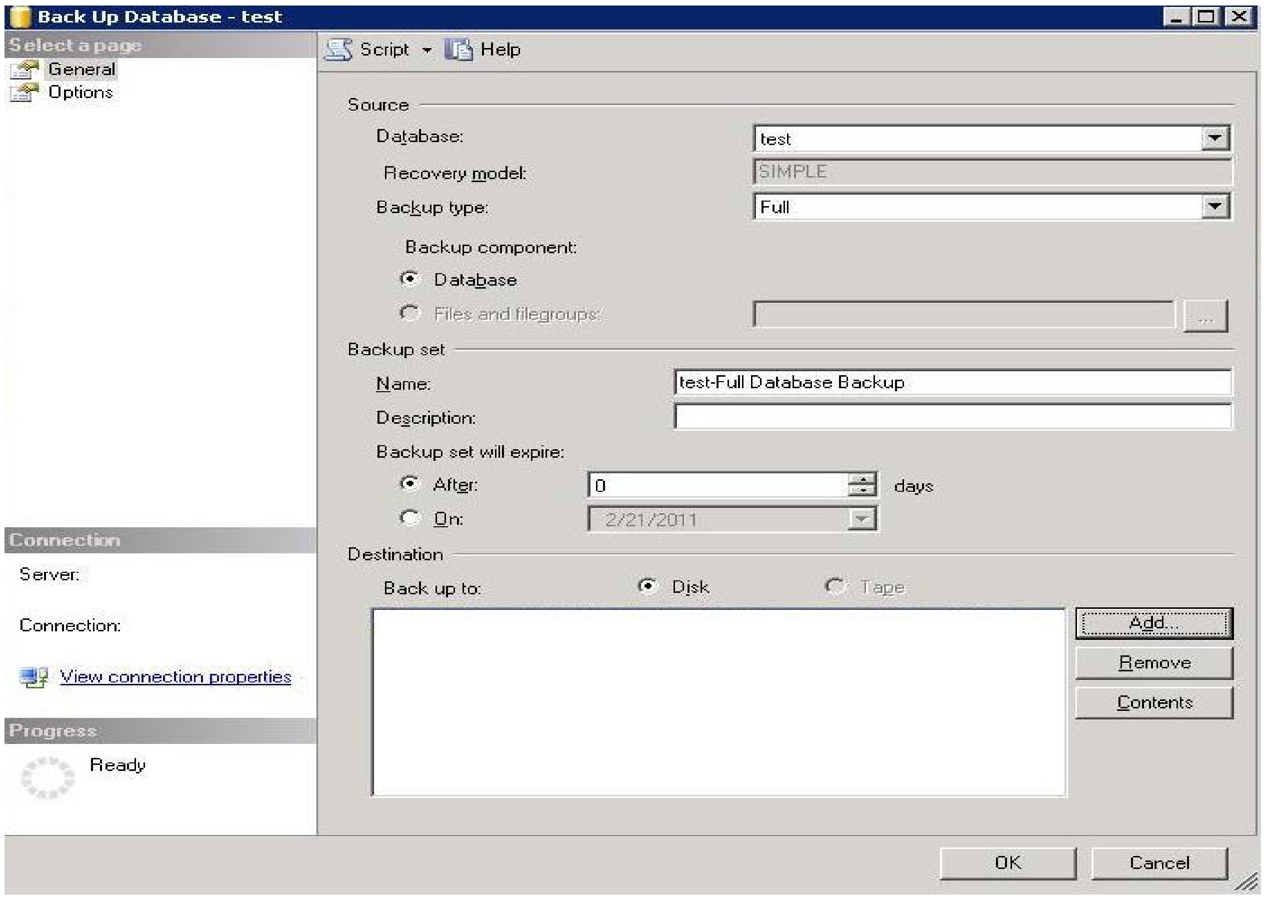
Step 1: Open your Microsoft SQL Server Management Studio.

Step 2: Using your Database Username and Password, simply login to your MS SQL server database.

Step 3: **Select the database >> Right-click >> Tasks >> Back Up** [as shown in the image below]:



Once you click on the "**Backup**" the following Backup Database window will appear.


Back Up Database - test

Select a page
General
Options

Script Help

Source
Database: test
Recovery model: SIMPLE
Backup type: Full
Backup component:
Database [✓]
Files and filegroups:

Backup set
Name: test-Full Database Backup
Description:
Backup set will expire:
After: [✓] 0 days
On: 2/21/2011

Destination
Back up to: Disk [✓] Tape

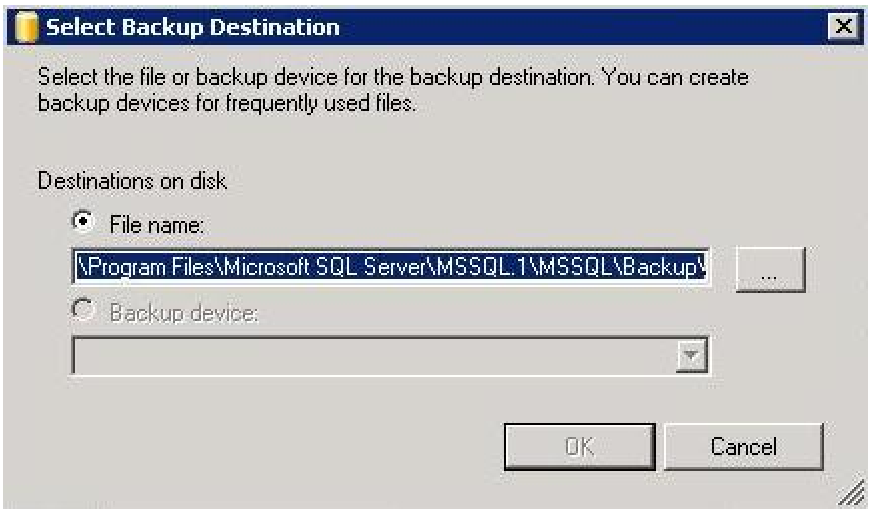
Connection
Server:
Connection:
View connection properties

Progress
Ready

Step 4: Select the following options:

1. Backup type: Full
2. Under Destination, Backup to: Disk

Step 5: Now, by clicking on the "**Add**" button, the following window will appear to select the path and file name for the database backup file [as shown in the image below]:



Select Backup Destination

Select the file or backup device for the backup destination. You can create backup devices for frequently used files.

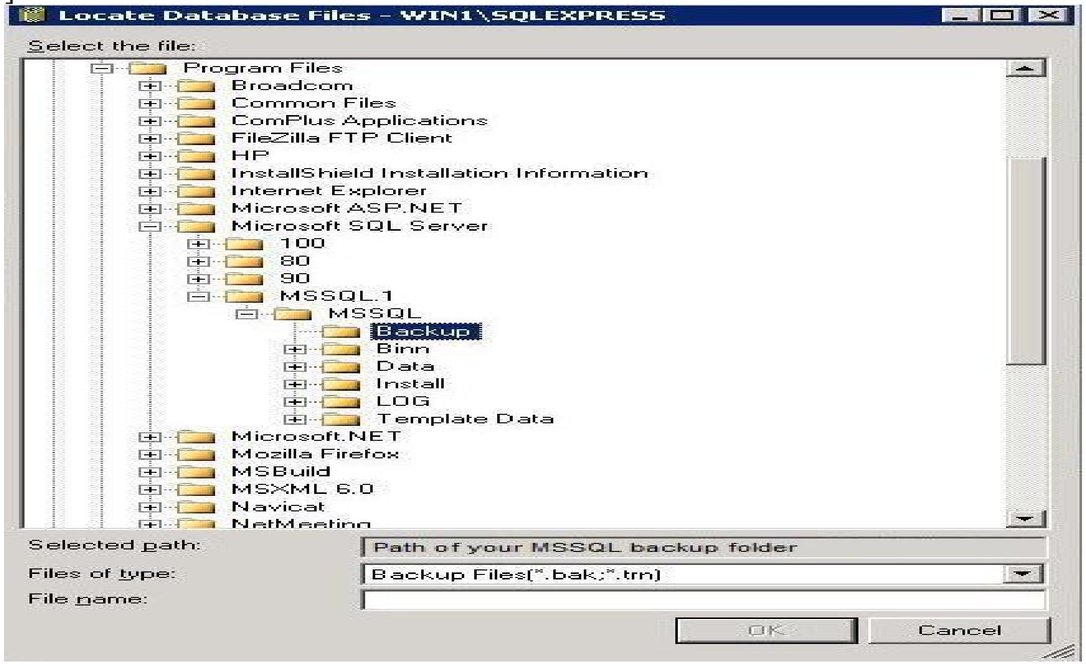
Destinations on disk

● File name:

\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Backup\

○ Backup device:

Step 6: Select the destination folder for the backup file and enter the "**File name**" with **.bak** extension [as shown in the image below]:



Locate Database Files - WIN1\SQLEXPRESS

Select the file:

```
Program Files
  Broadcom
  Common Files
  ComPlus Applications
  FileZilla FTP Client
  HP
  InstallShield Installation Information
  Internet Explorer
  Microsoft ASP.NET
  Microsoft SQL Server
    100
    80
    90
    MSSQL.1
      MSSQL
        Backup
        Binn
        Data
        Install
        LOG
        Template Data
  Microsoft.NET
  Mozilla Firefox
  MSBuild
  MSXML 6.0
  Navicat
  NetMeeting
```

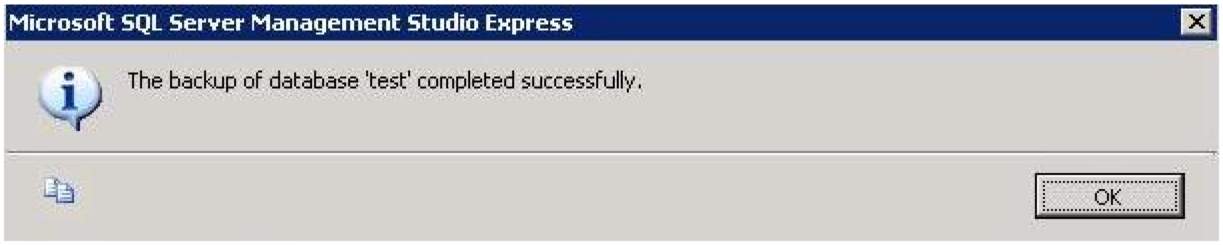
Selected path: Path of your MSSQL backup folder

Files of type: Backup Files(*.bak;*.trn)

File name:

Make sure you place your MS SQL database .bak file under the MSSQL backup folder.

Step 7: Hit the **OK** button to finish the backup of your MS SQL Server 2008 Database. Upon the successful completion of database backup, the following confirmation window will appear with a message "The backup of database "yourdatabasename" completed successfully. [as shown in the image below]:



Microsoft SQL Server Management Studio Express

The backup of database 'test' completed successfully.

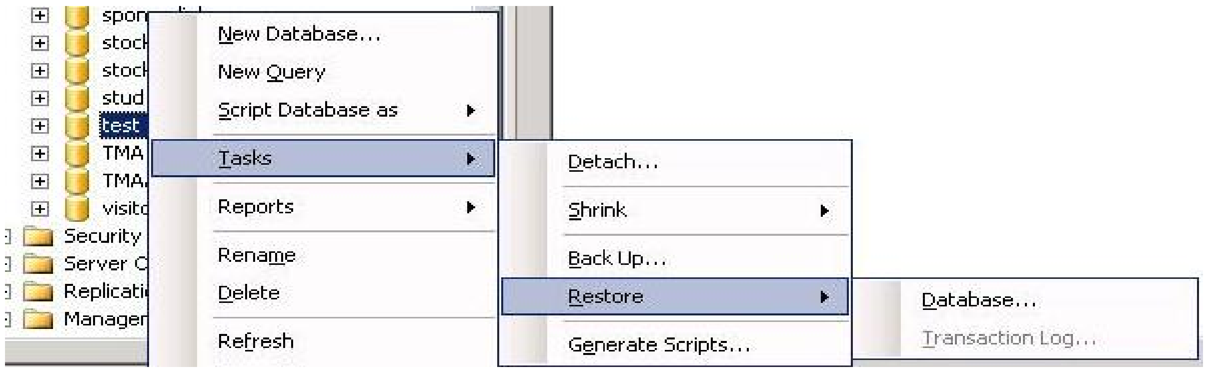
Following the above shown steps, you will be able to create a successful backup of your MS SQL Server 2008 Database into the desired folder.

How to Restore MS SQL Server 2008 Database Backup File?

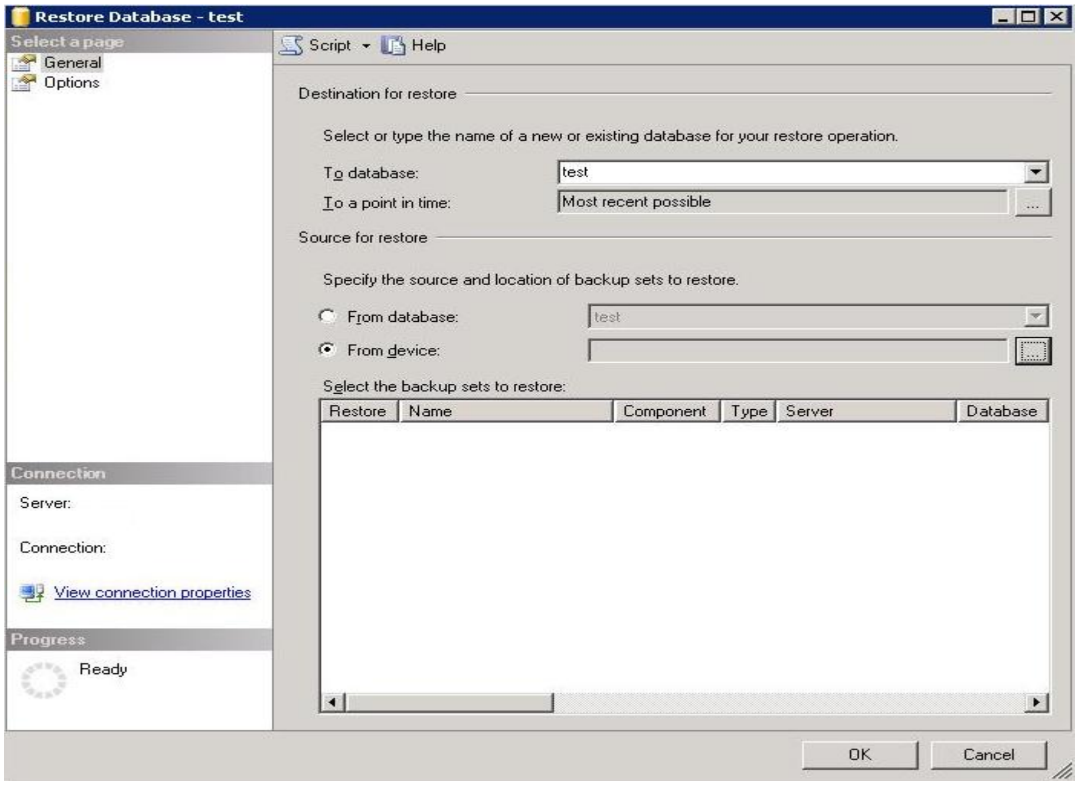
<u>To restore a database from a backup file, follow the steps shown below</u>:

Step 1: Open your Microsoft SQL Server Management Studio.

Step 2: **Select the database >> Right-click >> Tasks >> Restore >> Database** [as shown in the image below]:



Step 3: The following "**Restore Database**"windows will appear. Select "From device" mentioned under the "Source for restore" and click the button in front of that to specify the file location [as shown in the image below]:


Restore Database - test

Select a page
General
Options

Script Help

Destination for restore

Select or type the name of a new or existing database for your restore operation.

To database: test

To a point in time: Most recent possible

Source for restore

Specify the source and location of backup sets to restore.

From database: test

From device:

Select the backup sets to restore:

Restore	Name	Component	Type	Server	Database

Connection

Server:

Connection:

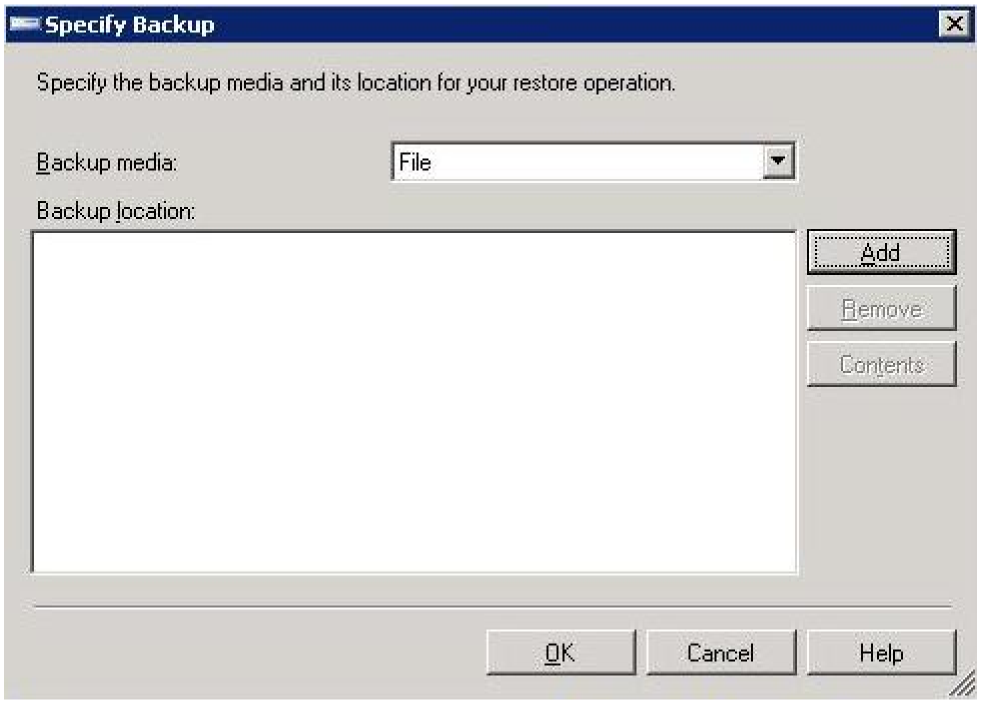
View connection properties

Progress

Ready

OK Cancel

Step 4: Select the option "Backup media as File" and click on the **Add** button to add the backup file location [as shown in the image below]:

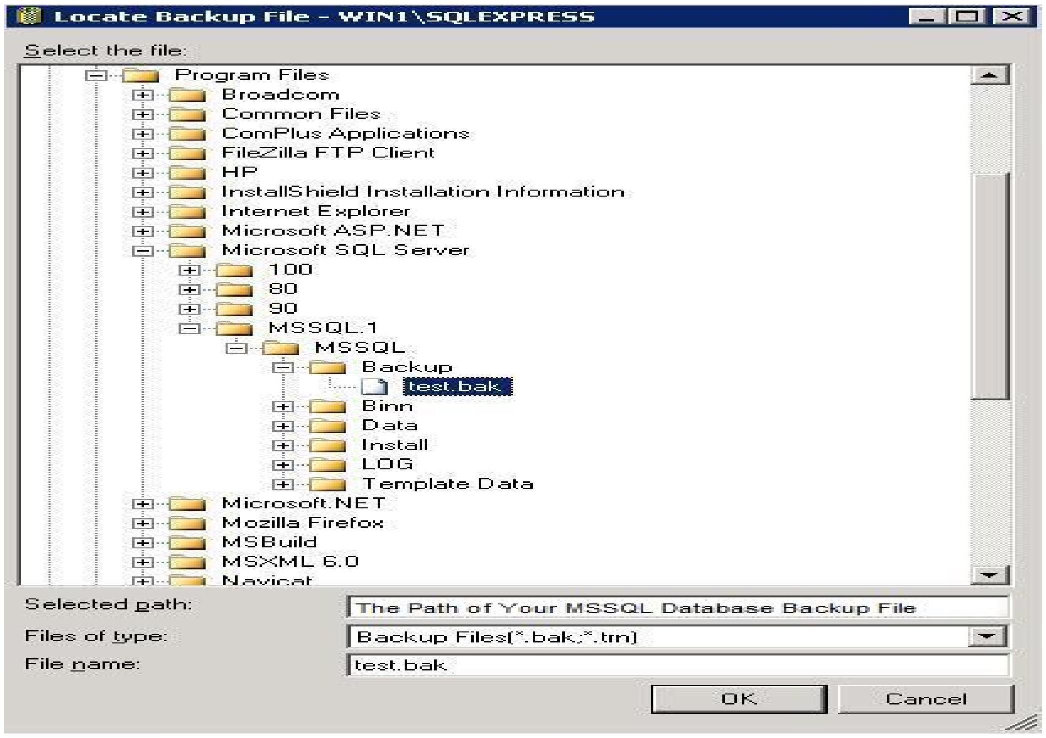


Specify the backup media and its location for your restore operation.

Backup media: File

Backup location:

Step 5: Select the backup file you wish to restore and Hit the **OK** button [as shown in the image below]:

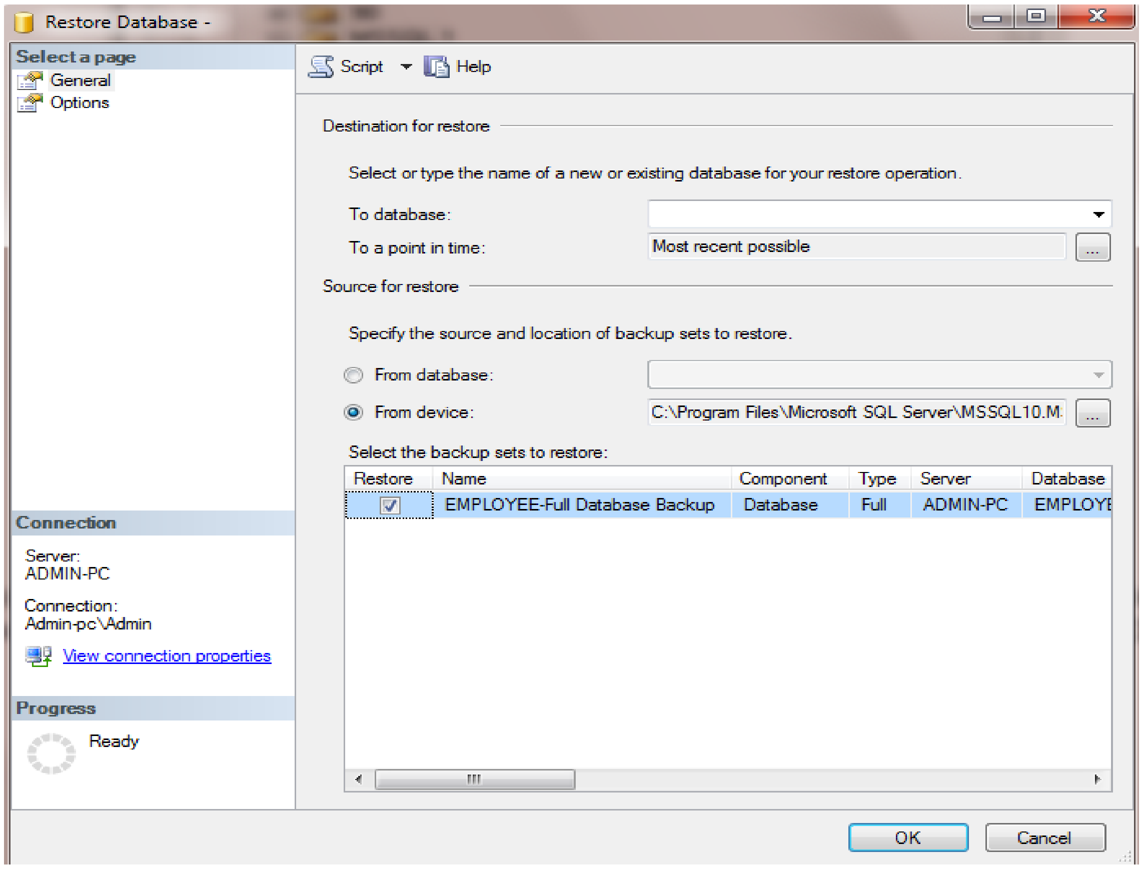


Selected path: The Path of Your MSSQL Database Backup File

Files of type: Backup Files(*.bak;*.trn)

File name: test.bak

You will get the confirmation windows with a message “The restoration of database “yourdatabasename” completed successfully.” Now you know the procedure of backing up and restoring MS SQL Server 2008 Database.


Restore Database -

Select a page
- General
- Options

Script · Help

Destination for restore

Select or type the name of a new or existing database for your restore operation.

To database:

To a point in time: Most recent possible

Source for restore

Specify the source and location of backup sets to restore.

From database:

From device: C:\Program Files\Microsoft SQL Server\MSSQL10.M

Select the backup sets to restore:

Restore	Name	Component	Type	Server	Database
✓	EMPLOYEE-Full Database Backup	Database	Full	ADMIN-PC	EMPLOYE

Connection

Server:
ADMIN-PC

Connection:
Admin-pc\Admin

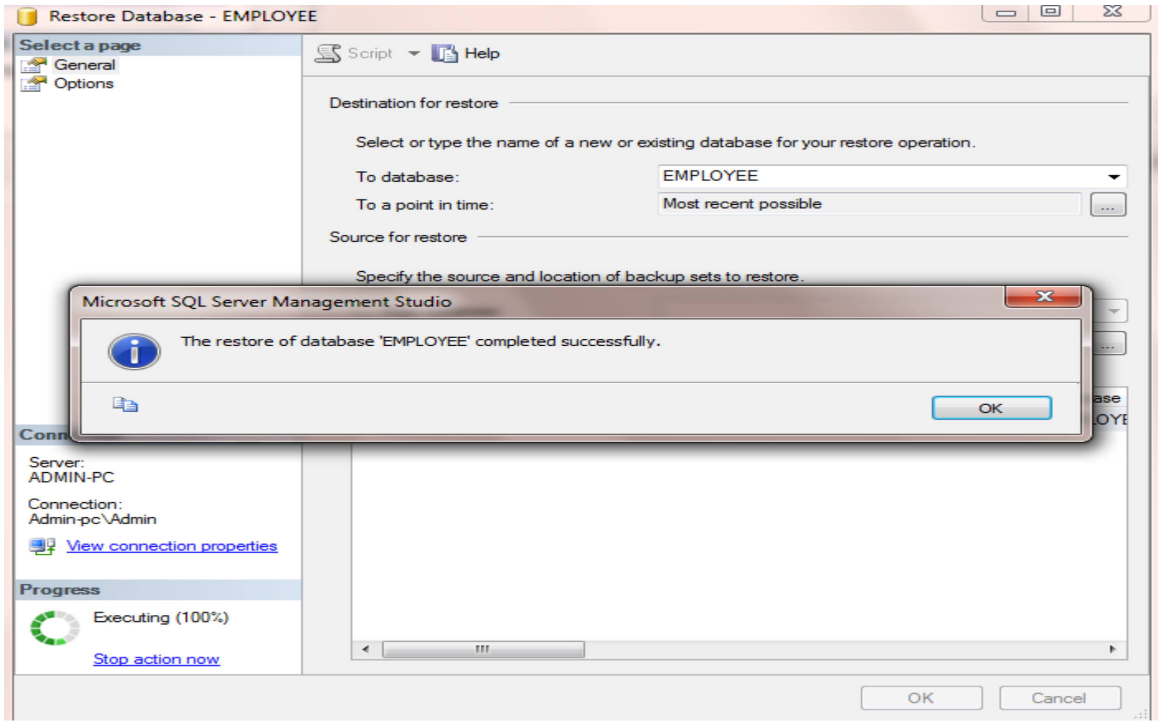
View connection properties

Progress

Ready

OK · Cancel

Step 6: Once this has completed, a pop-up will say the database has been restored successfully. Click **OK**.


Restore Database - EMPLOYEE

Select a page
- General
- Options

Script · Help

Destination for restore

Select or type the name of a new or existing database for your restore operation.

To database: EMPLOYEE

To a point in time: Most recent possible

Source for restore

Specify the source and location of backup sets to restore.

Microsoft SQL Server Management Studio

The restore of database 'EMPLOYEE' completed successfully.

OK

Server:
ADMIN-PC

Connection:
Admin-pc\Admin

View connection properties

Progress

Executing (100%)

Stop action now

OK · Cancel

You can create a back up and restore using SQL transaction code: use master

```
BACKUP DATABASE Employee TO DISK = 'D:\Employee.bak'
restore DATABASE Employee from disk= 'D:\Employee.bak'
```