Lab # 13

Index

In SQL Server Management Studio (SSMS), an index is a database object that improves the speed of data retrieval operations on a database table at the cost of additional space and decreased performance on data modification operations. There are two main types of indexes: clustered and non-clustered.

Clustered Index:

- A clustered index defines the order in which data is physically stored in a table.
- There can be only one clustered index per table because the data rows themselves can only be sorted in one order.
- When you create a clustered index on a table, the data rows are stored in the order of the clustered index key.
- Generally, it is recommended to create a clustered index on columns that are frequently searched for ranges of data or that are frequently used in JOIN clauses.

Non-Clustered Index:

- A non-clustered index does not affect the physical order of the table's rows.
- Instead, it creates a separate structure within the table that contains the indexed columns and a pointer to the actual row in the table.
- Multiple non-clustered indexes can be created on a single table.
- Non-clustered indexes are suitable for columns that are frequently used in WHERE clauses but are not suitable for columns that are frequently used in JOIN clauses.

To create an index using a command in SQL Server Management Studio (SSMS), you can use the CREATE INDEX statement. Here's a basic example:

```
CREATE INDEX IX_Employee_DepartmentID

ON dbo.Employee (DepartmentID);
```

This statement creates a non-clustered index named IX_Employee_DepartmentID on the DepartmentID column of the Employee table in the dbo schema.

If you want to create a clustered index, you can specify the CLUSTERED keyword:

```
CREATE CLUSTERED INDEX IX_Employee_EmployeeID
ON dbo.Employee (EmployeeID);
```

This statement creates a clustered index named IX_Employee_EmployeeID on the EmployeeID column of the Employee table.

You can also include additional options in the CREATE INDEX statement, such as specifying a fill factor or including other columns in the index. Here's an example that includes multiple columns in a non-clustered index:

```
CREATE NONCLUSTERED INDEX IX_Employee_Name_DepartmentID
ON dbo.Employee (Name ASC, DepartmentID DESC);
```

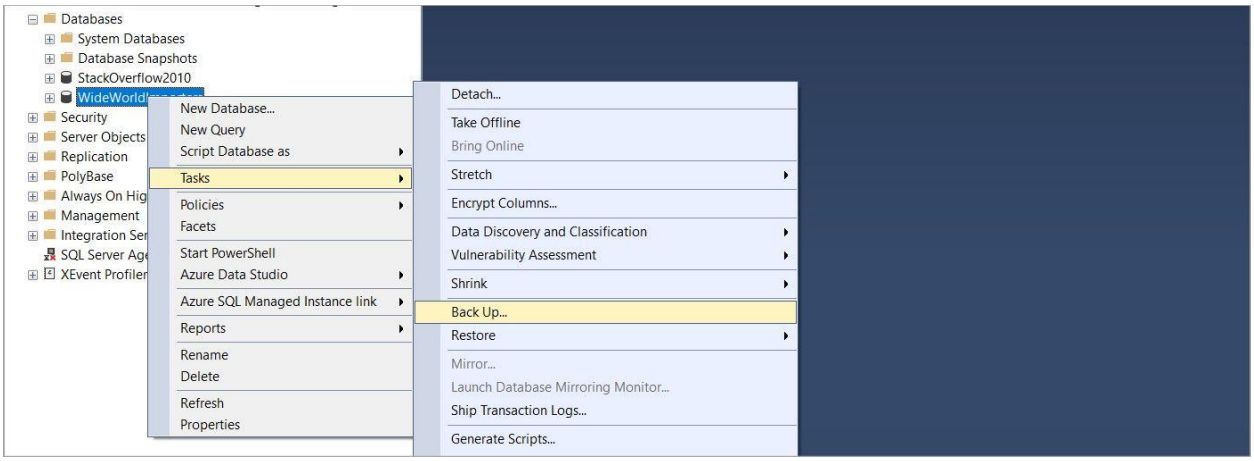
This statement creates a non-clustered index named IX_Employee_Name_DepartmentID on the Name column in ascending order (ASC) and the DepartmentID column in descending order (DESC).

Creating Backup and Restore Backup and Restore a Microsoft SQL Server Database With SSMS

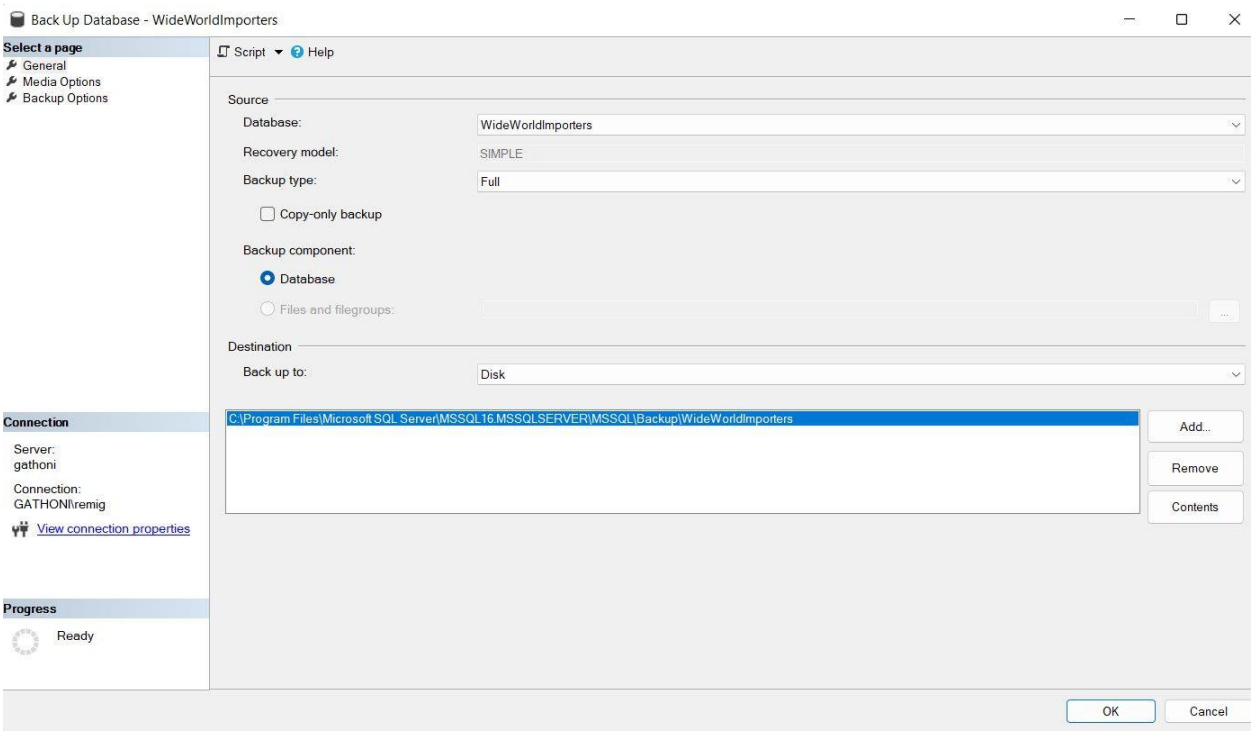
How to Back Up a Database Using SSMS

The SQL Server Management Studio (SSMS) lets you back up a database using its UI. To get started follow the steps outlined below.

1. Locate the database you want to back up under the **Databases** section and right-click on it.
2. In the database menu select **Tasks > Back Up**.



3. In the Backup Database dialog box, add the destination folder of the backup file. It should have a .bak file extension.



4. Click **OK** to start the backup process.

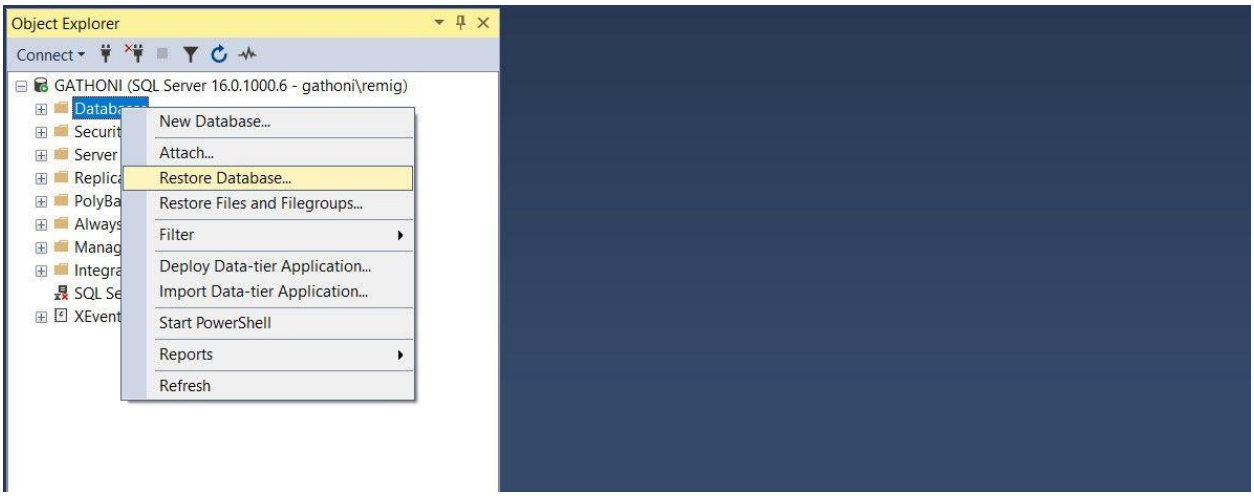
Instead of using the UI, you can also run the SQL query below to perform a full backup of the database.

BACKUP DATABASE WideWorldImporters **TO** DISK = 'C:\...\Backup\WideWorldImporters.bak'

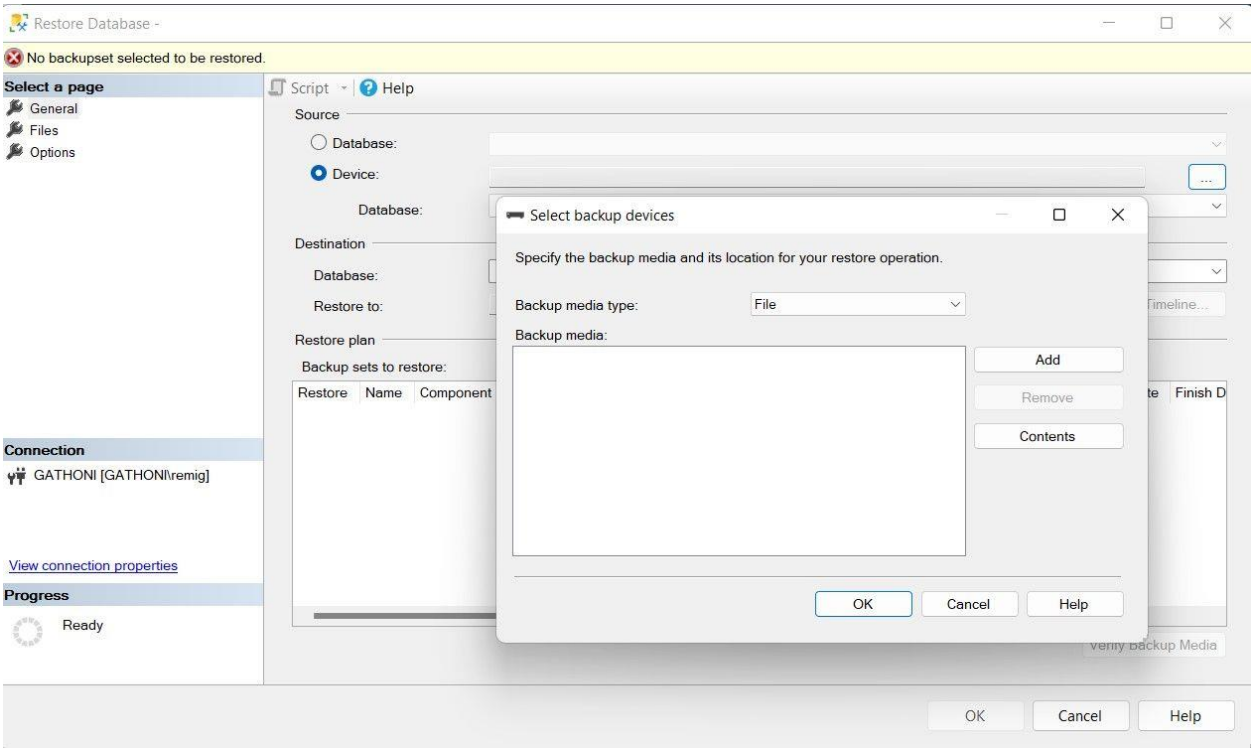
Restore an SQL Database Using SSMS

Follow the steps below to restore a backup database.

1. Launch SSMS and connect to the server.
2. In the left pane, right-click on the **Databases** node and select **Restore Database**.



3. Under **Source**, select **Devices** and click on the three dots button (**...**) to open the **Select backup devices** dialog box.



4. Click the **Add** button to choose the backup file then click **OK**.

Your database should now be restored.

Alternatively, you can use a SQL command to restore the database. Simply run the following query in the query window and specify the file location.

```
RESTORE DATABASE WideWorldImporters
```

```
FROM

DISK = 'C:\...\Backup\WideWorldImporters.bak'
```

The Importance of Database Backup and Recovery

Database backups and recovery are crucial in database maintenance. With recent backups, a DBA can restore a database to the last good copy if data is corrupted or attacked. This allows businesses or applications to continue running smoothly.

If you are going to maintain a database, you should ensure you can back up and restore an SQL database using SSMS.

Views

Views in SQL Server are virtual database objects that display data retrieved from one or more tables. They provide a way to simplify complex queries by encapsulating them into a reusable object. Views can be used to present specific subsets of data, hide complexity, and enforce security by restricting access to certain columns or rows.

Here's how views work in SQL Server:

1. Definition: A view is defined by a SQL query that retrieves data from one or more tables or other views.

2. Storage: Unlike tables, views do not store data themselves. Instead, they retrieve data dynamically from the underlying tables whenever they are queried.

3.Structure: When you query a view, SQL Server internally translates the query into a query against the underlying tables and returns the result set to the user.

4. **Updateability:** In some cases, views can be made updatable, allowing users to modify data through the view. However, this depends on various factors such as the complexity of the view and the underlying tables.

To create a view in SQL Server, you can use the `CREATE VIEW` statement. Here's the basic syntax:

```
CREATE VIEW view_name

AS

SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

Here's a breakdown of the syntax elements:

1. `view_name`: The name of the view you want to create.
2. `column1, column2, ...`: The columns you want the view to include. These can be columns from one or more tables, or they can be expressions or calculations.
3. `table_name`: The name of the table or tables from which you want to retrieve data.
4. `condition`: Optional. A condition to filter the rows retrieved from the table.

For example, let's say you have a `Employees` table with columns `EmployeeID`, `FirstName`, `LastName`, and `DepartmentID`, and you want to create a view that displays only the first and last names of employees in the IT department. You can create the view like this:

```
CREATE VIEW IT_Employees

AS

SELECT FirstName, LastName

FROM Employees

WHERE DepartmentID = 'IT';
```

Once the view is created, you can query it like a table:

```
SELECT * FROM IT_Employees;
```

This will return the first and last names of employees in the IT department.

Views are used for data retrieval and simplifying queries, while stored procedures are used for data processing and implementing complex business logic.