SQL - Backup Database

This tutorial will explain how we can take a backup of our database in MySQL as well as in MS SQL Server. It is very important to create backups of the databases because there might be a chance of data loss due to power surges or disk crashes etc. Overall, regular database backups are essential for ensuring the long-term availability of critical data.

Backup MySQL Database

MySQL **mysqldump** command can be used to take complete backup of a given database. This operation will be performed from command line and will require database user name and password, preferably admin privilege.

```
$ mysqldump -u username -p"password" -R testDB > testDB.sql
```

We are using the -p flag immediately followed by our password to connect to the database with no space between. The **-R** is required to tell **mysqldump** to copy stored procedures and functions along with the normal data from the database.

Depending on the database size, above command may take sometime to create a final output file testDB.sql. Once command is completed, you will have a complete database dump in testDB.sql file which you can keep safe anywhere you like. Later this file can be used to restore the database.

Restore MySQL Database

using **mysqladmin** prompt command as follows:

```
$ mysqladmin -u username -p"password" create tutorialsDB;
```

The next step is to import old database into new database as follwing:

```
$ mysql -u username -p"password" tutorialsDB < testDB.sql;
```

If you want to keep your database name same as the old one then you will have to drop old database and then re-create it before importing old data into this database, but make sure you don't have any data in this database which you do not want to loose.

Backup MS SQL Database

If you are working with MS SQL Server then to create a backup for an existing database, SQL provides us with a simple SQL **BACKUP DATABASE** command.

Syntax

Following is the syntax of the BACKUP DATABASE command in SQL −

```
BACKUP DATABASE database_name
TO DISK = 'filepath'
GO
```

Example

Following is an example to create a backup file for the database **testDB** on **D** drive.

```
SQL> BACKUP DATABASE testDB
TO DISK = 'D:\testDB.bak'
GO
```

privileges. You should also back up the database onto a different disk other than the actual database. Even if the disk crashes, we will not lose our backup file along with the database.

Output

When we execute the above query, the output is obtained as follows −

```
Processed 344 pages for database 'testDB', file 'testDB' on file 1.
Processed 2 pages for database 'testDB', file 'testDB_log' on file 1.
BACKUP DATABASE successfully processed 346 pages in 0.011 seconds (245.383
```

Backup with SQL DIFFERENTIAL Statement

The SQL Backup with a DIFFERENTIAL Statement is used to create a differential backup of the database. The differential backup contains only the changes made to the database since the last full backup. This type of backup is usually smaller in size compared to a full backup. Thus, it reduces the time to perform the backup.

Syntax

Following is the syntax for the backup database using DIFFERENTIAL Statement −

```
BACKUP DATABASE database_name
TO DISK = 'filepath'
WITH DIFFERENTIAL
GO
```

Example

Let us look at an example using the DIFFERENTIAL Statement below −

```
WITH DIFFERENTIAL
GO
```

Output

On executing the above query, the output is displayed as follows -

```
Processed 200 pages for database 'testDB', file 'testDB' on file 2.
Processed 2 pages for database 'testDB', file 'testDB_log' on file 2.
BACKUP DATABASE WITH DIFFERENTIAL successfully processed 202 pages in 0.011
```

Restore MS SQL Database

If you have a proper backup of an MS SQL database then youc an easily restore it when needed.

Syntax

Following is the syntax of the RESTORE DATABASE command in SQL -

```
RESTORE DATABASE database_name
FROM DISK = 'filepath'
[WITH REPLACE]
GO
```

Here **WITH REPLACE** option can be given if you want to overwrite the existing database.

Example

Following is an example to restore a database from a backup file **testDB.bak** available on **D** drive.

```
SQL> RESTORE DATABASE testDB
FROM DISK = 'D:\testDB.bak'
WITH REPLACE
GO
```