UNIT 2:
Backup and Recovery

Database Recovery

Database recovery techniques are used in database management systems (DBMS) to restore a database to a consistent state after a failure or error has occurred. The main goal of recovery techniques is to ensure data integrity and consistency and prevent data loss. There are mainly two types of recovery techniques used in DBMS:

Rollback/Undo Recovery Technique: The rollback/undo recovery technique is based on the principle of backing out or undoing the effects of a transaction that has not completed successfully due to a system failure or error. This technique is accomplished by undoing the changes made by the transaction using the log records stored in the transaction log. The transaction log contains a record of all the transactions that have been performed on the database. The system uses the log records to undo the changes made by the failed transaction and restore the database to its previous state.

Commit/Redo Recovery Technique: The commit/redo recovery technique is based on the principle of reapplying the changes made by a transaction that has been completed successfully to the database. This technique is accomplished by using the log records stored in the transaction log to redo the changes made by the transaction that was in progress at the time of the failure or error. The system uses the log records to reapply the changes made by the transaction and restore the database to its most recent consistent state.

In addition to these two techniques, there is also a third technique called checkpoint recovery. Checkpoint recovery is a technique used to reduce the recovery time by periodically saving the state of the database in a checkpoint file. In the event of a failure, the system can use the checkpoint file to restore the database to the most recent consistent state before the failure occurred, rather than going through the entire log to recover the database.

Overall, recovery techniques are essential to ensure data consistency and availability in DBMS, and each technique has its own advantages and limitations that must be considered in the design of a recovery system

Database systems, like any other computer system, are subject to failures but the data stored in them must be available as and when required. When a database fails it must possess the facilities for fast recovery. It must also have atomicity i.e. either transaction are completed successfully and committed (the effect is recorded permanently in the database) or the transaction should have no effect on the database. There are both automatic and non-automatic ways for both, backing up of data and recovery from any failure situations. The techniques used to recover the lost data due to system crashes, transaction errors, viruses, catastrophic failure, incorrect commands execution, etc. are database recovery techniques. So to prevent data loss recovery techniques based on deferred update and immediate update or backing up data can be used. Recovery techniques are heavily dependent upon the existence of a special file known as a **system log**. It contains information about the start and end of each transaction and any updates which occur during the **transaction**. The log keeps track of all transaction operations that affect the values of database items. This information is needed to recover from transaction failure.

- The log is kept on disk start_transaction(T): This log entry records that transaction T starts the execution.
- read_item(T, X): This log entry records that transaction T reads the value of database item X.
- write_item(T, X, old_value, new_value): This log entry records that transaction T changes the value of the database item X from old_value to new_value. The old value is sometimes known as a before an image of X, and the new value is known as an afterimage of X.
- commit(T): This log entry records that transaction T has completed all accesses to the database successfully and its effect can be committed (recorded permanently) to the database.
- abort(T): This records that transaction T has been aborted.
- checkpoint: Checkpoint is a mechanism where all the previous logs are removed from the system and stored permanently in a storage disk. Checkpoint declares a point before which the DBMS was in a consistent state, and all the transactions were committed.

A transaction T reaches its **commit** point when all its operations that access the database have been executed successfully i.e. the transaction has reached the point at which it will not **abort** (terminate without completing). Once committed, the transaction is permanently recorded in the database. Commitment always involves writing a commit entry to the log and writing the log to disk. At the time of a system crash, item is searched back in the log for all transactions T that have written a start_transaction(T) entry into the log but have not written a commit(T) entry yet; these transactions may have to be rolled back to undo their effect on the database during the recovery process.

- **Undoing –** If a transaction crashes, then the recovery manager may undo transactions i.e. reverse the operations of a transaction. This involves examining a transaction for the log entry write_item(T, x, old_value, new_value) and set the value of item x in the database to old-value. There are two major techniques for recovery from non-catastrophic transaction failures: deferred updates and immediate updates.
- **Deferred update –** This technique does not physically update the database on disk until a transaction has reached its commit point. Before reaching commit, all transaction updates are recorded in the local transaction workspace. If a transaction fails before reaching its commit point, it will not have changed the database in any way so UNDO is not needed. It may be necessary to REDO the effect of the operations that are recorded in the local transaction workspace, because their effect may not yet have been written in the database. Hence, a deferred update is also known as the **No-undo/redo algorithm**
- **Immediate update –** In the immediate update, the database may be updated by some operations of a transaction before the transaction reaches its commit point. However, these operations are recorded in a log on disk before they are applied to the database, making recovery still possible. If a transaction fails to reach its commit point, the effect of its operation must be undone i.e. the transaction must be rolled back hence we require both undo and redo. This technique is known as **undo/redo algorithm.**
- **Caching/Buffering –** In this one or more disk pages that include data items to be updated are cached into main memory buffers and then updated in memory before being written back to disk. A collection of in-memory buffers called the DBMS cache is kept under the control of DBMS for holding these buffers. A directory is used to keep track of which database items are in the buffer. A dirty bit is associated with each buffer, which is 0 if the buffer is not modified else 1 if modified.
- **Shadow paging –** It provides atomicity and durability. A directory with n entries is constructed, where the ith entry points to the ith database page on the link. When a transaction began executing the current directory is copied into a shadow directory. When a page is to be modified, a shadow page is allocated in which changes are made and when it is ready to become durable, all pages that refer to the original are updated to refer new replacement page.
- **Backward Recovery** – The term “Rollback ” and “UNDO” can also refer to backward recovery. When a backup of the data is not available and previous modifications need to be undone, this technique can be helpful. With the backward recovery

method, unused modifications are removed and the database is returned to its prior condition. All adjustments made during the previous traction are reversed during the backward recovery. In another word, it reprocesses valid transactions and undoes the erroneous database updates.

- **Forward Recovery** – "Roll forward "and "REDO" refers to forwarding recovery. When a database needs to be updated with all changes verified, this forward recovery technique is helpful.
  Some failed transactions in this database are applied to the database to roll those modifications forward. In another word, the database is restored using preserved data and valid transactions counted by their past saves.

Some of the backup techniques are as follows :

- **Full database backup –** In this full database including data and database, Meta information needed to restore the whole database, including full-text catalogs are backed up in a predefined time series.
- **Differential backup –** It stores only the data changes that have occurred since the last full database backup. When some data has changed many times since last full database backup, a differential backup stores the most recent version of the changed data. For this first, we need to restore a full database backup.
- **Transaction log backup –** In this, all events that have occurred in the database, like a record of every single statement executed is backed up. It is the backup of transaction log entries and contains all transactions that had happened to the database. Through this, the database can be recovered to a specific point in time. It is even possible to perform a backup from a transaction log if the data files are destroyed and not even a single committed transaction is lost.

MySQL Export and Import Database

How can we Export and Import database in MySQL?

Database export and import in MySQL is a process of moving data from one place to another place. Export and import are useful methods for backing up essential data or transferring our data between different versions. For example, we have a contact book database that is essential for our business. It is necessary to keep it in a secure place. So we need to export it in a safe place, and whenever it lost from the original location, we can restore it using import options.

In MySQL, we can export and import database in mainly two ways:

1. Command Line Tool

2. MySQL Workbench

Let us discuss both ways of database exporting and importing in detail.

MySQL Export Database

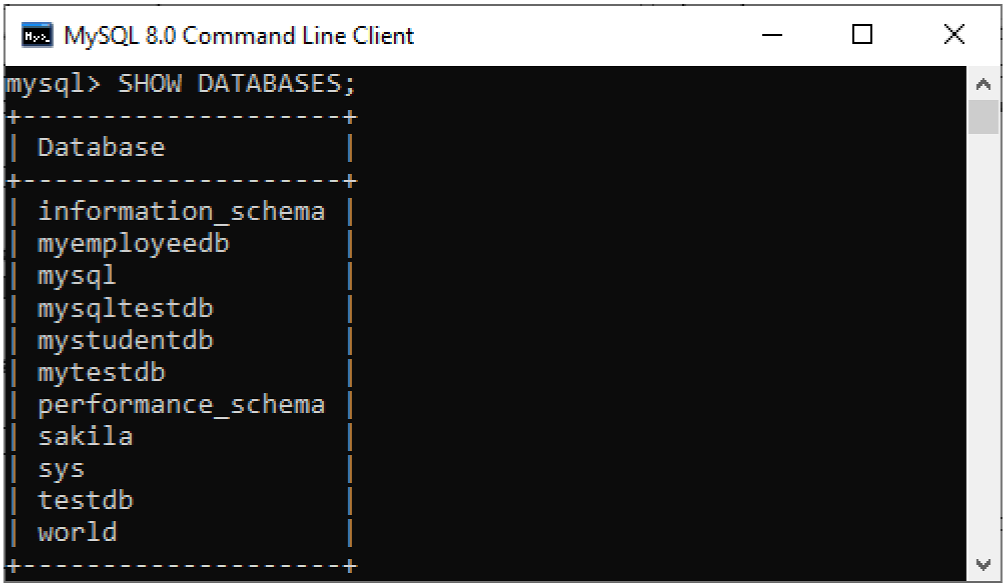
Command Line Tool

Here, we are going to see database exporting, including tables using the command-line tool. So open the command-line tool using the username and password and perform the following steps:

Step 1: Execute the below statement to show all databases available on MySQL Server.

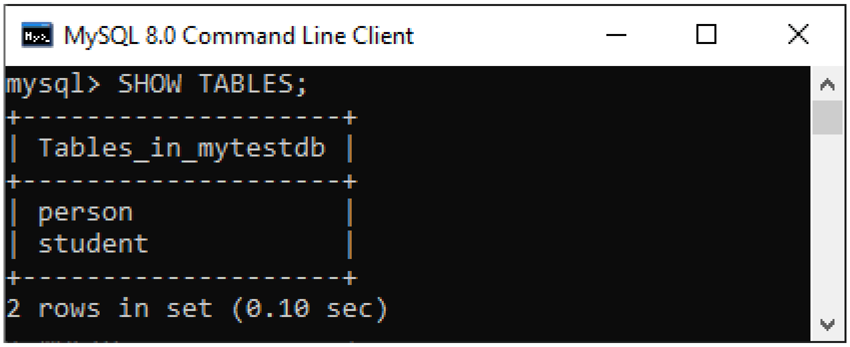
```
1. mysql> SHOW DATABASES;
```

We will get the following output:



```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| myemployeedb       |
| mysql              |
| mysqltestdb        |
| mystudentdb        |
| mytestdb           |
| performance_schema |
| sakila             |
| sys                |
| testdb             |
| world              |
+--------------------+
```

Step 2: From the above databases, we are going to take a database named **"mytestdb"** that contains the following table:



```
mysql> SHOW TABLES;
+--------------------+
| Tables_in_mytestdb |
+--------------------+
| person             |
| student            |
+--------------------+
2 rows in set (0.10 sec)
```

Step 3: Access the command line on the computer where the database is stored. You can open a DOS or terminal window to access the command line if you have physical access to the computer. **For example**, if we have installed the MySQL in the **C folder**, copy the following folder and paste it in our DOS command. Now, press **Enter key**.

```
1. C:\Users\javatpoint> CD C:\Program Files\MySQL\MySQL Server 8.0\bin
```

Step 4: Now, we will use the **mysqldump** tool to export the database. This tool uses the login credentials of MySQL user for the operation. The following command is used to export the database in your desired place.

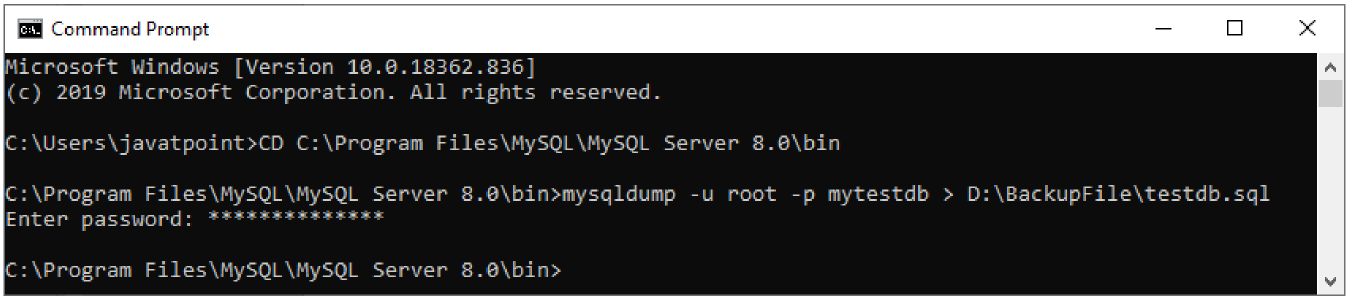
```
1. $ mysqldump -u username -p database_name > desiredplace\dbname.sql
```

-u: It ensures MySQL username will be followed.
Username: It is the name of a user to which we can log in to the database.
-p: It ensures the password associated with the username.
database_name: It is the name of a database that we want to export.
>: It ensures the output location.
desiredplace\dbname.sql: The desired_place is the folder where we want to export, and dbname.sql is the file with which the output will be saved.

In the below statement, we will give a username: **root**, database name: **mytestdb**, folder name: **BackupFile**, and output database name: **testdb.sql.** Now, press the Enter key.

```
1. mysqldump -u root -p mytestdb > D:\BackupFile\testdb.sql
```

Step 5: After pressing the Enter key, it will ask the password of the username we have specified. Then, press the Enter key again. It will create the backup file with a **.sql suffix** in the specified location. We will get the output as below that means the database exported successfully:



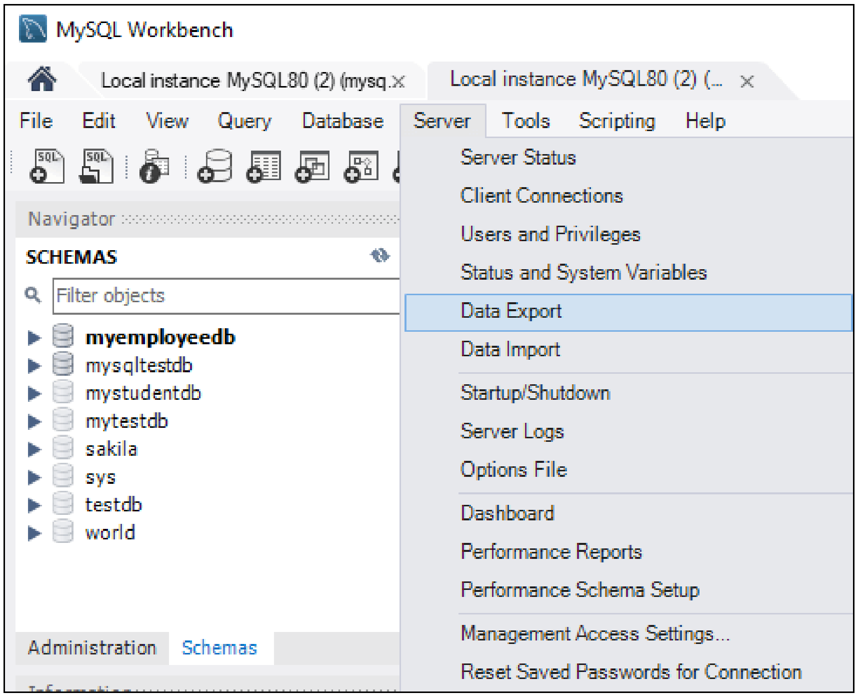
```
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\javatpoint>CD C:\Program Files\MySQL\MySQL Server 8.0\bin

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysqldump -u root -p mytestdb > D:\BackupFile\testdb.sql
Enter password: **************

C:\Program Files\MySQL\MySQL Server 8.0\bin>
```

Finally, we can verify the export database, including tables in the specified folder.

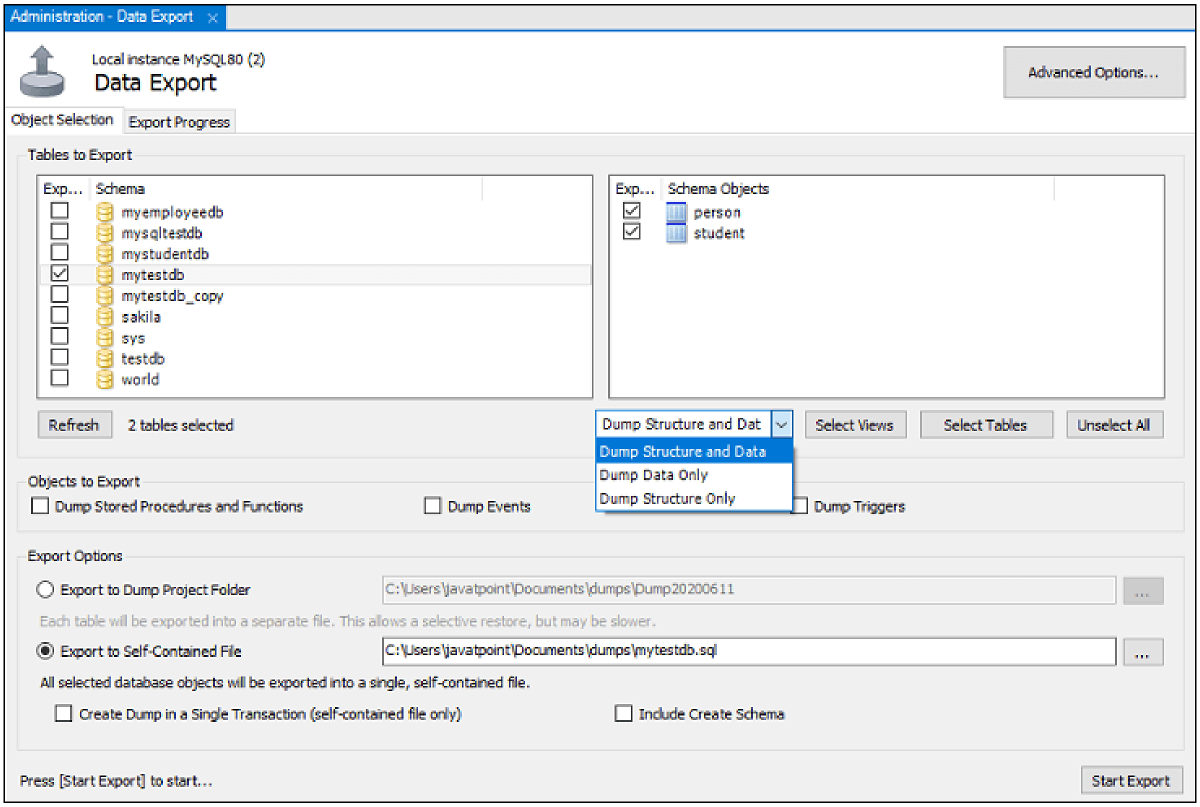
MySQL Workbench

Here, we are going to see database exporting, including tables using MySQL Workbench. So open the workbench and type the password for the username.

Step 1: Go to the Menu bar and click on the Server. A popup screen appears, then select the **Data Export** option, as shown in this screen. It will open a new window of data export settings and options.



Step 2: Select any database that you want. It will also display corresponding tables in the left section of the window. Here, we are going to select the **mytestdb** database. We can also choose multiple database checkboxes to include the database in the Export file. Similarly, we can select multiple tables.



Local instance MySQL80 (2)

Data Export

Tables to Export

Exp...	Schema
□	myemployeedb
□	mysqltestdb
□	mystudentdb
☑	mytestdb
□	mytestdb_copy
□	sakila
□	sys
□	testdb
□	world

Exp...	Schema Objects
☑	person
☑	student

2 tables selected

Dump Structure and Dat

- Dump Structure and Data
- Dump Data Only
- Dump Structure Only

Objects to Export

□ Dump Stored Procedures and Functions □ Dump Events □ Dump Triggers

Export Options

○ Export to Dump Project Folder C:\Users\javatpoint\Documents\dumps\Dump20200611

Each table will be exported into a separate file. This allows a selective restore, but may be slower.

● Export to Self-Contained File C:\Users\javatpoint\Documents\dumps\mytestdb.sql

All selected database objects will be exported into a single, self-contained file.

□ Create Dump in a Single Transaction (self-contained file only) □ Include Create Schema

Press [Start Export] to start...

Step 3: After selecting the database, including all tables, go to the drop-down setting, and select any of the available options.

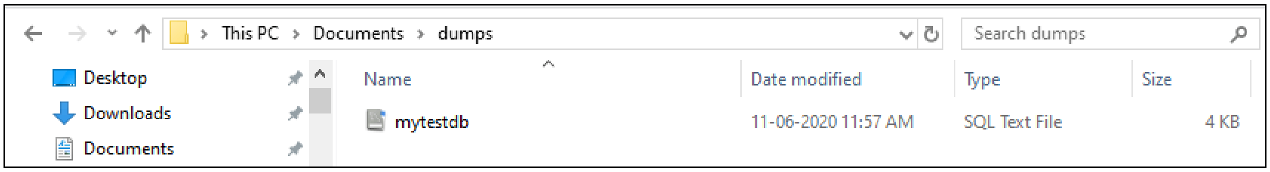
- **Dump Data and Structure:** It will save both table structure and data rows.
- **Dump Data Only:** It will save only the inserted rows in the tables.
- **Dump Structure Only:** It will save only the table structure, which are database columns and data types defined by us.

Step 4: In the Export option, we can see two radio buttons that are explained below.

- **Export to Dump Project Folder:** It will save all the tables as separate SQL files under one folder. It will be useful when you import or restore the export file one by one table.
- **Export to Self-Contained File:** It will store all the databases and tables in a single SQL file. It is a good option when you want to import all the databases, tables, and data rows using a single SQL file.

We can select the export path of our choice. Here, I will keep the default setting.

Step 5: Click the Start Export button, which displays the progress bar and log. Now, we can verify the export files in the Document folder in our system.



Name	Date modified	Type	Size
mytestdb	11-06-2020 11:57 AM	SQL Text File	4 KB

MySQL Import Database

Command Line Tool

Here, we are going to see database importing, including tables using the command-line tool. So open the command-line tool using the username and password and perform the following steps:

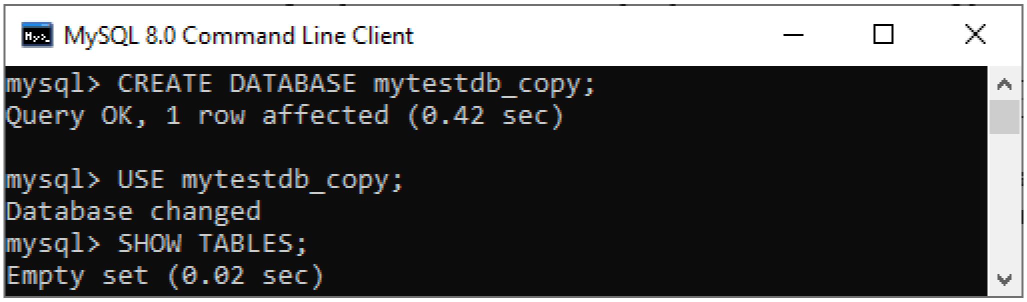
Step 1: Create a blank database named **"mytestdb_copy"** by using the below statement:

```
mysql> CREATE DATABASE mytestdb_copy;
```

Step 2: Next, we need to execute the below commands to verify that it does not have any table.

```
mysql> USE mytestdb_copy;
mysql> SHOW TABLES;
```

We can see the following output:



```
mysql> CREATE DATABASE mytestdb_copy;
Query OK, 1 row affected (0.42 sec)

mysql> USE mytestdb_copy;
Database changed
mysql> SHOW TABLES;
Empty set (0.02 sec)
```

Step 3: Access the command line on the computer where the database is stored. You can open a DOS or terminal window to access the command line if you have physical access to the computer. **For example**, if we have installed the MySQL in the **C folder**, copy the following folder link and paste it in your DOS command. Now, press Enter key.

1. C:\Users\javatpoint> CD C:\Program Files\MySQL\MySQL Server 8.0\bin

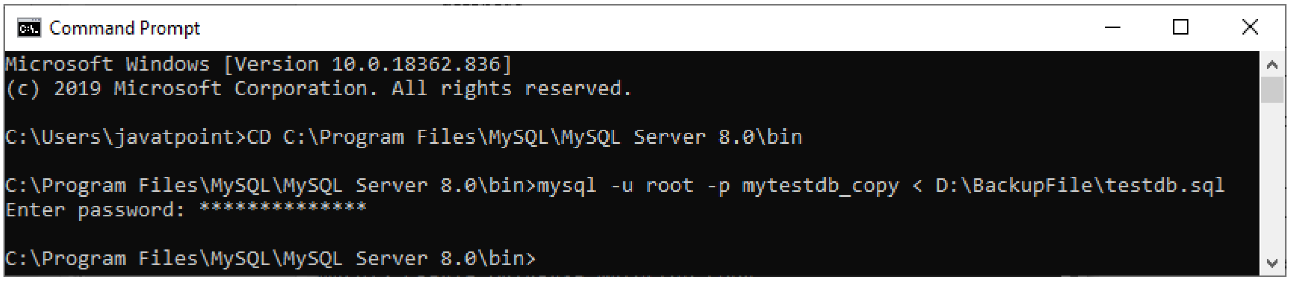
Step 4: Next, we will use the following command to import the dump file in your desired database.

1. $ mysql -u username -p database_name < desiredplace\dbname.sql

In the below statement, we will give a username: **root**, database name: **mytestdb_copy**, location of backup file: **BackupFile**, and input database name: **testdb.sql**. Now, press Enter key.

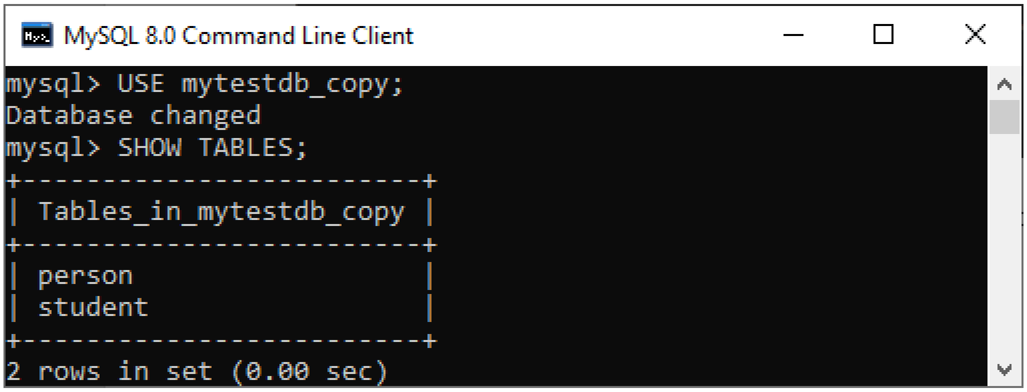
1. $ mysql -u root -p mytestdb_copy < D:\BackupFile\testdb.sql

Step 5: After pressing the Enter key, it will ask the password of the username we have specified. Then, press the Enter key again. If the command executed correctly, it does not display any content on the screen. We will get the below screen that means the database imported successfully:



```
Microsoft Windows [Version 10.0.18362.836]
(c) 2019 Microsoft Corporation. All rights reserved.

C:\Users\javatpoint>CD C:\Program Files\MySQL\MySQL Server 8.0\bin

C:\Program Files\MySQL\MySQL Server 8.0\bin>mysql -u root -p mytestdb_copy < D:\BackupFile\testdb.sql
Enter password: **************

C:\Program Files\MySQL\MySQL Server 8.0\bin>
```

Step 6: Open the MySQL client tool again and execute the below command to verify the database. In the output, we can see that this database now contains two tables:

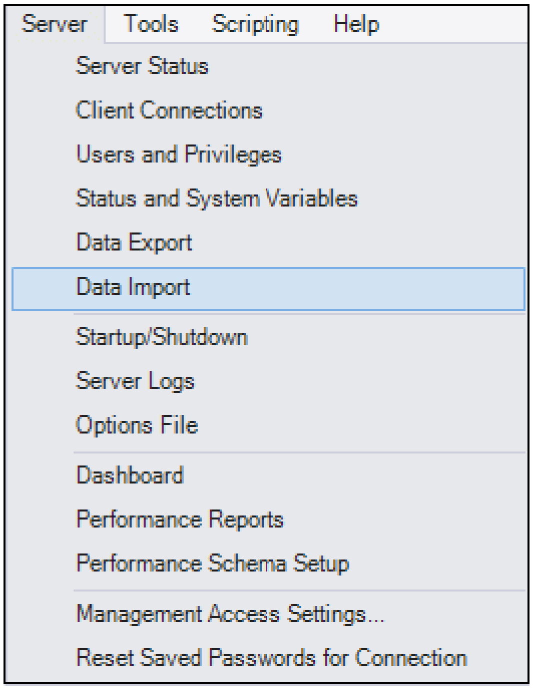


```
mysql> USE mytestdb_copy;
Database changed
mysql> SHOW TABLES;
+-------------------------+
| Tables_in_mytestdb_copy |
+-------------------------+
| person                  |
| student                 |
+-------------------------+
2 rows in set (0.00 sec)
```

MySQL Workbench

Here, we are going to see database importing using MySQL Workbench. So open the workbench and enter the password for the username.

Step 1: Navigate to the Menu bar and click on the Server. A popup screen appears, then select the **Data Import** option, as shown in this screen. It will open a new window of data import settings and options.

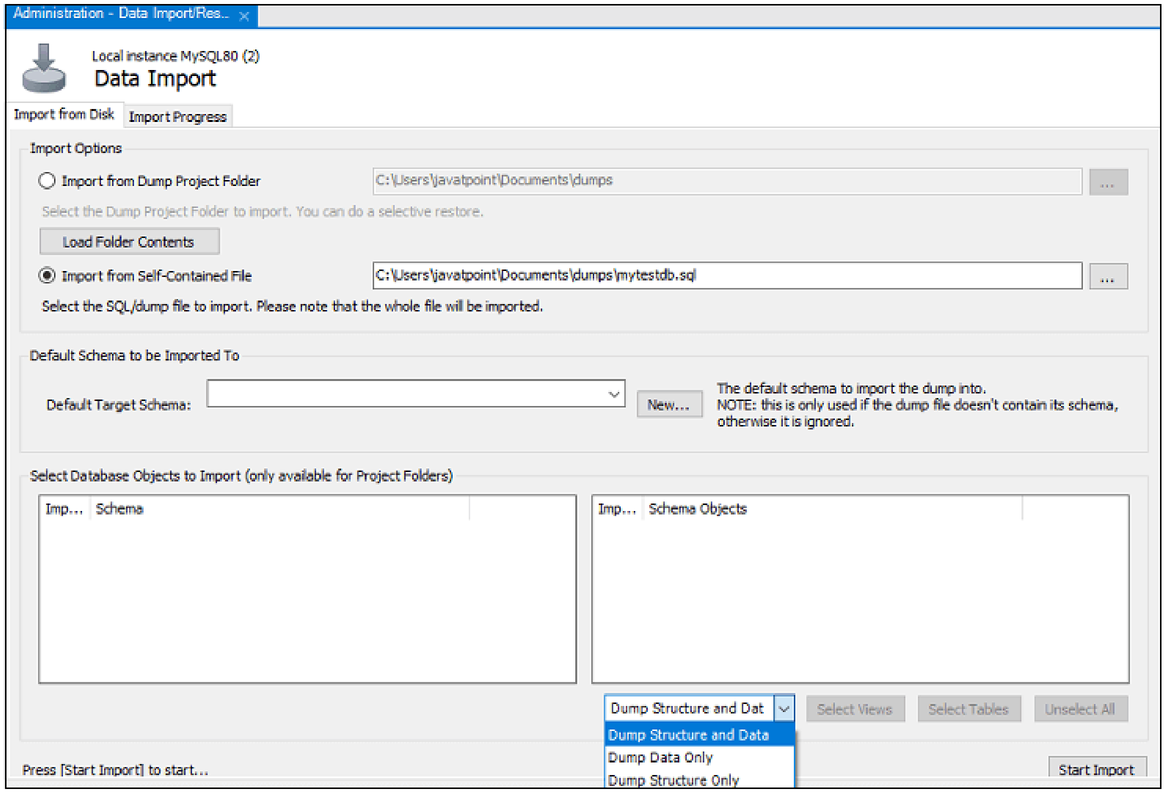


Step 2: In the screen, we can see the two radio options to import databases and tables, which are given below:

- Import from Dump Project Folder
- Import by using Self-Contained File

Step 3: Here, we will select **'Import by using Self-Contained File'** and choose the desired database for importing from the Data Import option.

Step 4: Choose the **'Dump Structure and Data'** option and click the **Start Import** button to import the databases and tables from the backup file. The following screen explains all the steps clearly:



○ Import from Dump Project Folder: C:\Users\javatpoint\Documents\dumps

Select the Dump Project Folder to import. You can do a selective restore.

● Import from Self-Contained File: C:\Users\javatpoint\Documents\dumps\mytestdb.sql

Select the SQL/dump file to import. Please note that the whole file will be imported.

Default Target Schema:

The default schema to import the dump into.
NOTE: this is only used if the dump file doesn't contain its schema, otherwise it is ignored.

Dump Structure and Dat

Press [Start Import] to start...

Step 5: Now, navigate to the Schema under the navigator option on the left side of the workbench window and refresh it to see the currently imported database or table.

Hardware Protection and Type of Hardware Protection

In this article, we are going to learn about hardware protection and its types. So first, lets take a look at the type of hardware which is used in a computer system. We know that a computer system consists of hardware components like processor, monitor, RAM and many more. The important thing is, that the operating system ensures that these devices are not directly accessible by the user.

Basically, hardware protection is divided into 3 categories: CPU protection, Memory Protection, and I/O protection. These are explained as follows:

1. CPU Protection:

CPU protection ensures that, a process does not monopolize the CPU indefinetely, as it would prevent other processes from being executed. Each process should get a limited time, so that every process gets time to execute it's instructions. To address this, a timer is used to limit the amount of time, which a process can occupy from the CPU. After the timer expires, a signal is sent to the process for relenquishing the CPU. Hence one process cannot hold the CPU forever.

2. Memory Protection:
In memory protection, we are talking about that situation when two or more processes are in memory and one process may access the other process memory. To prevent this situation we use two registers which are known as:
1. Base register
2. Limit register
So basically **Base register** store the starting address of program and limit register store the size of the process. This is done to ensure that whenver a process wants to access the memory, the OS can check that – Is the memory area which the process wants to access is previliged to be accessed by that process or not.

3. I/O Protection:
With I/O protection, an OS ensures that following can be never done by a processes:

1. **Termination I/O of other process** – This means one process should not be able to terminate I/O operation of othe processes.
2. **View I/O of other process** – One process should not be able to access the data being read/written by other processes from/to the Disk(s).
3. **Giving priority to a particular process I/O** – No process must be able to priorotize itself or other processes which are doing I/O operations, over other processes.

What is data redundancy?

Data redundancy refers to the practice of keeping data in two or more places within a database or data storage system. Data redundancy ensures an organization can provide continued operations or services in the event something happens to its data -- for example, in the case of data corruption or data loss. The concept applies to areas such as databases, computer memory and file storage systems.

Data redundancy can occur within an organization intentionally or accidentally. If done intentionally, the same data is kept in different locations with the organization making a conscious effort to protect it and ensure its consistency. This data is often used for backups or disaster recovery.

If carried out by accident, duplicate data may cause data inconsistencies. Even though data redundancy can help minimize the chance of data loss, redundancy issues can affect larger data sets. For example, data that is stored in several places takes up valuable storage space and makes it difficult for the organization to identify which data they should access or update.

The word *redundant* can also be used as an independent technical term to refer to the following:

1. Computer or network system components that are installed to back up primary resources in case they fail.
2. Redundant information that is unneeded or duplicated.
3. Redundant bits or extra binary digits that are generated and moved with a data transfer to ensure that no bits were lost during the data transfer.
4. Redundant data that protects a storage array against data loss in the event of a hard disk failure.

How does data redundancy work?

Data needs to be stored in two or more places for it to be considered redundant. If the primary data becomes corrupted, or if the hard drive the data is on fails, then the extra set of data provides a fail-safe the organization can shift to.

The redundant data can be either a whole copy of the original data or select pieces of data. Keeping select pieces of data enables an organization to reconstruct lost or damaged data. Hard drives with copies of data are stored in an array, so if something happens to the original data, the array can kick in with little to no downtime. In addition, redundancy measures can be accomplished through backups or RAID systems.

Benefits and drawbacks of data redundancy

Data redundancy has benefits or risks depending on the implementation. Potential benefits include the following:

- **Helps protect data.** When data cannot be accessed, redundant data can help replace or rebuild missing data.

- **Data accuracy.** Hosting multiple locations for the same data means that a data management system can evaluate any differences, meaning data is assured to be accurate.
- **Access speed.** Some locations for data may be easier to access than others for an organization that spans different physical areas. A person within an organization may access data from redundant sources to have faster access to the same data.