Unit-2

MySQL

Unit-2: MySQL (Syllabus)

1. SQL and MySQL
2. Basic DDL and DML statements
3. joins and views and MongoDB database
4. Importing-exporting and querying data
5. creating and manipulating documents
6. CRUD operation
7. indexing and aggregation pipeline

MySQL:

MySQL is an open-source relational database management system (RDBMS) that uses Structured Query Language (SQL). It's one of the most popular databases in the world, known for its reliability, scalability, and ease of use. MySQL is commonly used in web development alongside scripting languages like PHP, Python, and Ruby to create dynamic websites and web applications.

Key features of MySQL include:

1. **Relational Database:** MySQL organizes data into tables with rows and columns, following the relational model.
2. **SQL Support:** MySQL supports SQL for querying and managing data. This includes commands for creating, retrieving, updating, and deleting data (CRUD operations), as well as more complex operations like joins and subqueries.
3. **Concurrency Control:** MySQL manages concurrent access to the database, ensuring that multiple users can work with the data simultaneously without interfering with each other.
4. **Transactions:** MySQL supports transactions, which allow multiple database operations to be grouped together as a single unit. This ensures data integrity and consistency, even in the presence of errors or failures.

5. Replication and Clustering: MySQL supports replication, allowing data to be copied across multiple servers for redundancy and scalability. It also supports clustering for high availability and load balancing.

6. Security: MySQL provides various security features, including user authentication, access control, encryption, and auditing, to protect sensitive data from unauthorized access or modification.

7. Community and Support: MySQL has a large and active community of developers and users who contribute to its development, provide support, and share knowledge through forums, blogs, and documentation.

1. SQL and MySQL

What is SQL?

SQL is an abbreviation used for Structured Query Language. This language is used to maintain and handle the database. It is defined as the standard language to work with databases by ANSI (American National Standards Institute) guidelines. Several other languages allow the user to implement and modify databases. There are only a few minor changes in the syntax of another language. Most queries that can be used to retrieve, add or manipulate data are based on the syntax of the SQL.

Structured Query Language was developed at IBM. The user can implement SQL queries that enable the user to add, alter, delete, or update the data in the database. It uses English-like language to write easy-to-understand queries for the databases. The user can implement SQL to handle several relational database management systems such as MySQL, SQL Server, and Oracle. The user can type the code and query in SQL to create or modify the existing databases and develop the schemas for the database.

What is SQL Server?

❖ SQL Server is an example of a relational database management system that allows the user to store tabular data and perform various operations such as insertion, deletion, and manipulation of the data stored in the databases. It was developed by Microsoft organization.

❖ It acts as a platform that makes storing, managing, and retrieving data from different applications and services easier. The user can implement queries using SQL; this language enables the user to interact with the databases. Users can use simple queries to create, update or delete databases, tables, and views.

❖ The user can perform several operations on data, making it relatively easier to read the data from the databases. These functions include querying, filtering, sorting, and merging the data.

❖ SQL Server also includes features such as high availability, increased scalability, and security. This is why SQL Server is a popular choice among businesses of all sizes.

❖ It is used to implement databases in web applications and perform data analysis.

What is MySQL?

MySQL is a relational database management system developed by MySQL AB software company. It was launched in 1995. Sun Microsystems later acquired the software company. Sun Microsystems is now known as the Oracle Corporation. MySQL comprises two words My, after the name of the co-founder's daughter, and SQL, which is short for the structured query language.

It is one of the earliest relational database management systems made available to the general public. Since it was an open-source management system, several variants are available for MySQL. These variants may have different features, but the basic syntax for all the variations is kept the same. The primary code for the management system was developed and written in C and C++. It was supported by all primary operating systems (OS). MySQL is the primary component used with an open-source web application stack known as LAMP. LAMP refers to Linux, Apache, MySQL, and PHP/Perl/Python.

How Do You Use SQL with MySQL?

- SQL is a query language that is used to manage relational databases. It is used used to implement queries for the management system. It is a standard language that enables users to handle and modify the data stored in a relational database.
- MySQL is not a programming language; instead, it is one of the most frequently used open-source relational database management systems that provide an interface for the user to implement SQL queries on the database. The user can implement SQL queries in the MySQL application, but the user must have the MySQL application preinstalled in the system. It should also have client software to provide support for SQL.
- Suppose the user wants to communicate with MySQL by implementing SQL commands. In that case, the user can either use the MySQL command line or a graphical user interface of the application to implement the SQL queries. MySQL Workbench is a GUI application allowing users to interact with SQL. It enables them to use SQL commands such as SELECT, INSERT, UPDATE, and DELETE to extract, insert, manipulate, or remove data in MySQL databases.

What is the Difference Between SQL and MySQL?

Since now, you have formed an understanding of all the necessary concepts related to SQL and MySQL. Let us discuss the difference between SQL and MySQL.

SQL	MySQL
SQL stands for Structured Query Language. It is a query language that allows users to manage relational database management systems.	MySQL is an RDBMS that can implement SQL queries to manipulate the data.
SQL allows the user to implement queries and work with relational database systems.	It enables the user to manage, store, edit, or remove the data and arrange it in an organized manner.
It does not provide support for any connectors. It is a simple programming language for writing queries.	MySQL provides users with several inbuilt applications. These tools include MySQL Workbench, which enables the user to create, design or build a database.
SQL is a standard language that has a fixed format. It does not require to require regular changes.	It has several variants and frequent updates are introduced by the developers to add new features.

SQL provides support for only one storage engine.	MySQL enables the user to use multiple storage engines. The user can also use additional plug-in storage that makes it more flexible.
SQL does not permit other processors or its own binaries to modify the data while executing the query.	MySQL is comparatively less secure as it enables other processors to modify the data files even when performing the execution.

What are SQL and MySQL Used For?

SQL is a standard programming language that allows users to manage and manipulate relational databases. SQL is primarily used to develop an application for industries such as finance and healthcare that handles a large amount of data.

In contrast to SQL, MySQL is an open-source relational database management system used to manage databases in RDBMS. It enables the user to store, handle and retrieve data in web applications. MySQL is often implemented with PHP and other programming languages, enabling the user to develop dynamic web applications.

What are the Similarities Between SQL and MySQL?

- Knowing SQL and MySQL is essential if you are planning a career in data management. SQL is used to implement queries in several database management systems. It is used to manage relational databases.
- MySQL is an open-source relational database management system enabling users to write queries in SQL. However, these technologies are different in numerous aspects. There are some similarities between SQL and MySQL.
- Primarily, both these technologies support using different data types such as numeric, string, date, and boolean.
- Secondly, it also enables the user to manipulate the data. The user can use DML commands such as SELECT, INSERT, UPDATE, and DELETE.
- Lastly, both technologies function in similar methods. The syntax for the queries is the same in both SQL and MySQL. It uses the same keywords to implement the query. The keywords include SELECT, FROM, WHERE, and ORDER BY. The user can learn either one of the technology, and it is easier to switch to another.

How to Choose What is Best to Use Between SQL and MySQL?

When choosing between SQL and MySQL, the user needs to remember that SQL is a query programming language, meaning that SQL defines the syntax to implement the query where.

In contrast, MySQL is a relational database management system that enables the user to execute queries using SQL. MySQL is a relational database management system that allows users to execute queries in SQL.

It is a standard language used to add, manipulate or remove data from relational databases, whereas MySQL can be considered a platform to implement the queries.

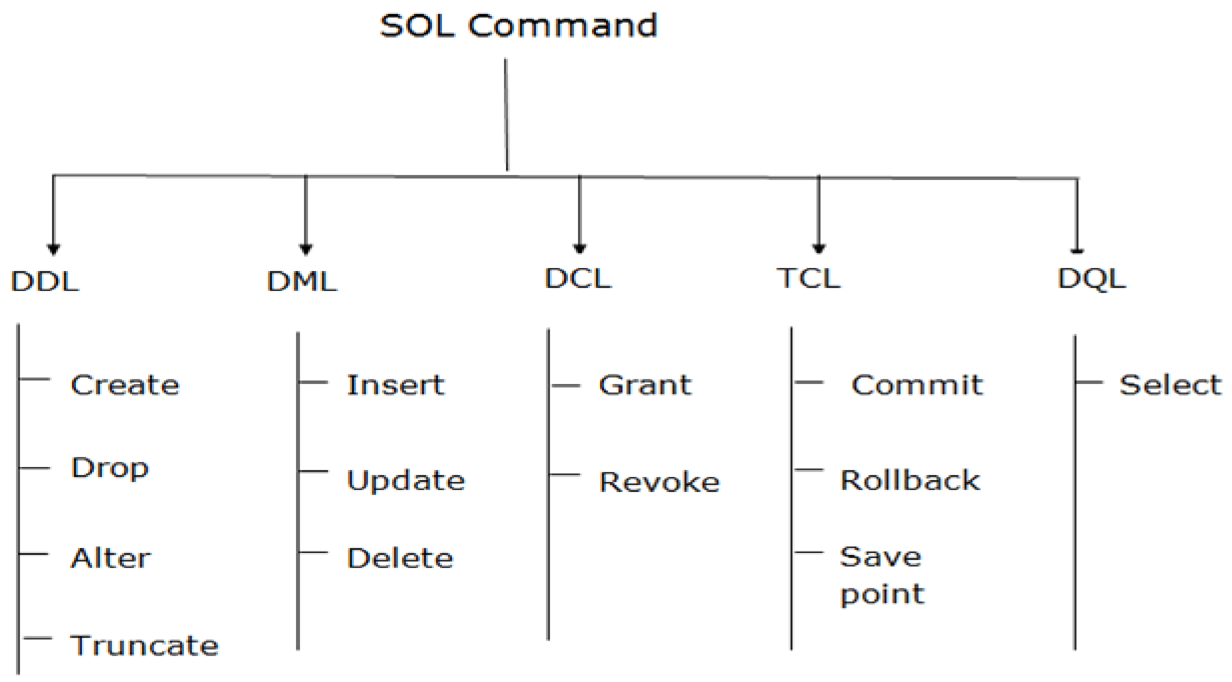
2. Basic DDL and DML statements

SQL Commands

- SQL commands are instructions. It is used to communicate with the database. It is also used to perform specific tasks, functions, and queries of data.
- SQL can perform various tasks like create a table, add data to tables, drop the table, modify the table, set permission for users.

Types of SQL Commands

There are five types of SQL commands: DDL, DML, DCL, TCL, and DQL.



➢Data Definition Language (DDL)

DDL changes the structure of the table like creating a table, deleting a table, altering a table, etc.

All the command of DDL are auto-committed that means it permanently save all the changes in the database.

Here are some commands that come under DDL:

- CREATE
- ALTER
- DROP
- TRUNCATE

a. CREATE

It is used to create a new table in the database.

Syntax: CREATE TABLE TABLE_NAME (COLUMN_NAME DATATYPES[,....]);

Example:

```
CREATE TABLE EMPLOYEE(Name VARCHAR2(20), Email VARCHAR2(100), DOB DATE);
```

b. DROP: It is used to delete both the structure and record stored in the table.

Syntax

DROP TABLE table_name;

Example

DROP TABLE EMPLOYEE;

c. ALTER: It is used to alter the structure of the database. This change could be either to modify the characteristics of an existing attribute or probably to add a new attribute.

Syntax:

To add a new column in the table

ALTER TABLE table_name ADD column_name COLUMN-definition;

To modify existing column in the table:

ALTER TABLE table_name MODIFY(column_definitions....);

EXAMPLE

ALTER TABLE STU_DETAILS ADD(ADDRESS VARCHAR2(20));

ALTER TABLE STU_DETAILS MODIFY (NAME VARCHAR2(20));

d. TRUNCATE: It is used to delete all the rows from the table and free the space containing the table.

Syntax:

```
TRUNCATE TABLE table_name;
```

Example:

```
TRUNCATE TABLE EMPLOYEE;
```

➢Data Manipulation Language

- DML commands are used to modify the database. It is responsible for all form of changes in the database.
- The command of DML is not auto-committed that means it can't permanently save all the changes in the database. They can be rollback.

Here are some commands that come under DML:

- INSERT
- UPDATE
- DELETE

a. **INSERT**: The INSERT statement is a SQL query. It is used to insert data into the row of a table.

Syntax:

INSERT INTO TABLE_NAME

(col1, col2, col3,.... col N)

VALUES (value1, value2, value3, ... valueN);

Or

INSERT INTO TABLE_NAME

VALUES (value1, value2, value3, ... valueN);

For example:

INSERT INTO javatpoint (Author, Subject) VALUES ("Sonoo", "DBMS");

b. **UPDATE**: This command is used to update or modify the value of a column in the table.

Syntax:

UPDATE table_name SET [column_name1= value1,...column_nameN = valueN] [WHERE CONDITION]

For example:

- UPDATE students
- SET User_Name = 'Sonoo'
- WHERE Student_Id = '3’

c. **DELETE**: It is used to remove one or more row from a table.

Syntax:

DELETE FROM table_name [WHERE condition];

For example:

DELETE FROM javatpoint

WHERE Author="Sonoo";

3.Joins and views and MongoDB database

JOINS

MySQL JOINS are used with SELECT statement. It is used to retrieve data from multiple tables. It is performed whenever you need to fetch records from two or more tables.

There are three types of MySQL joins:

- MySQL INNER JOIN (or sometimes called simple join)
- MySQL LEFT OUTER JOIN (or sometimes called LEFT JOIN)
- MySQL RIGHT OUTER JOIN (or sometimes called RIGHT JOIN)

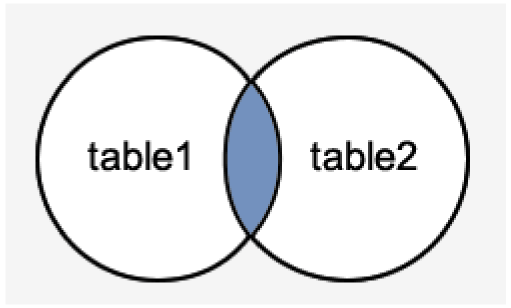
1) MySQL Inner JOIN (Simple Join)

The MySQL INNER JOIN is used to return all rows from multiple tables where the join condition is satisfied. It is the most common type of join.

Syntax:

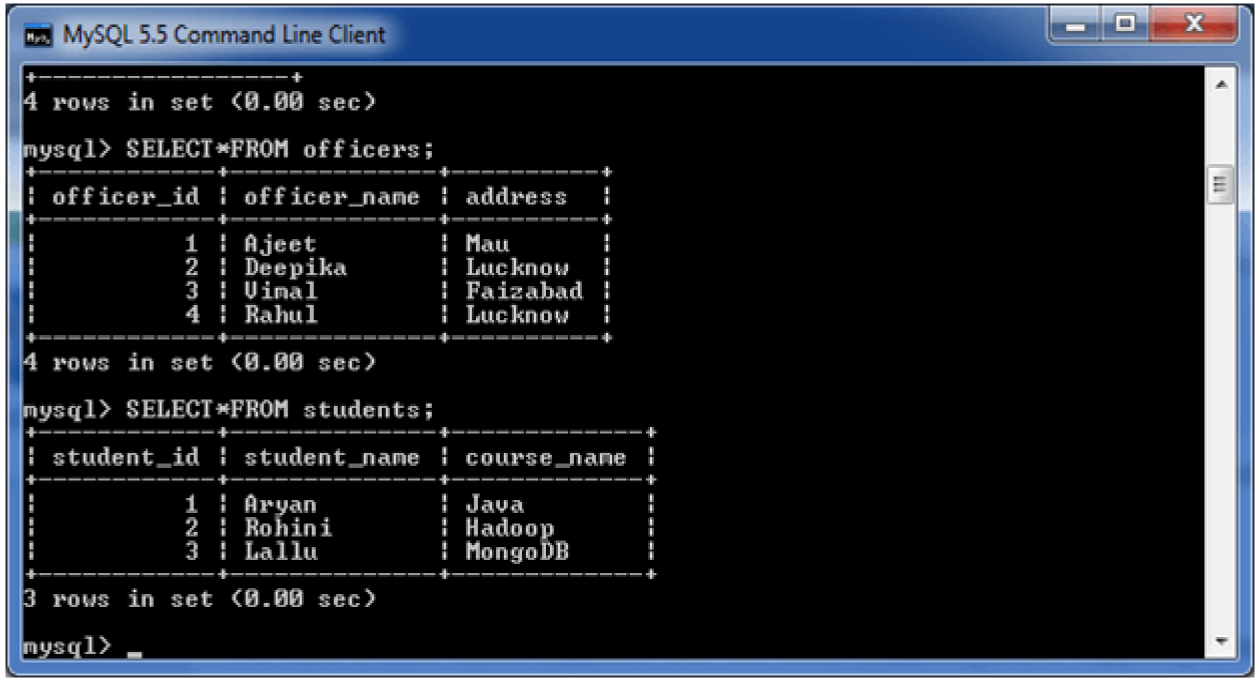
```
SELECT columns
FROM table1
INNER JOIN table2
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.



```
+------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

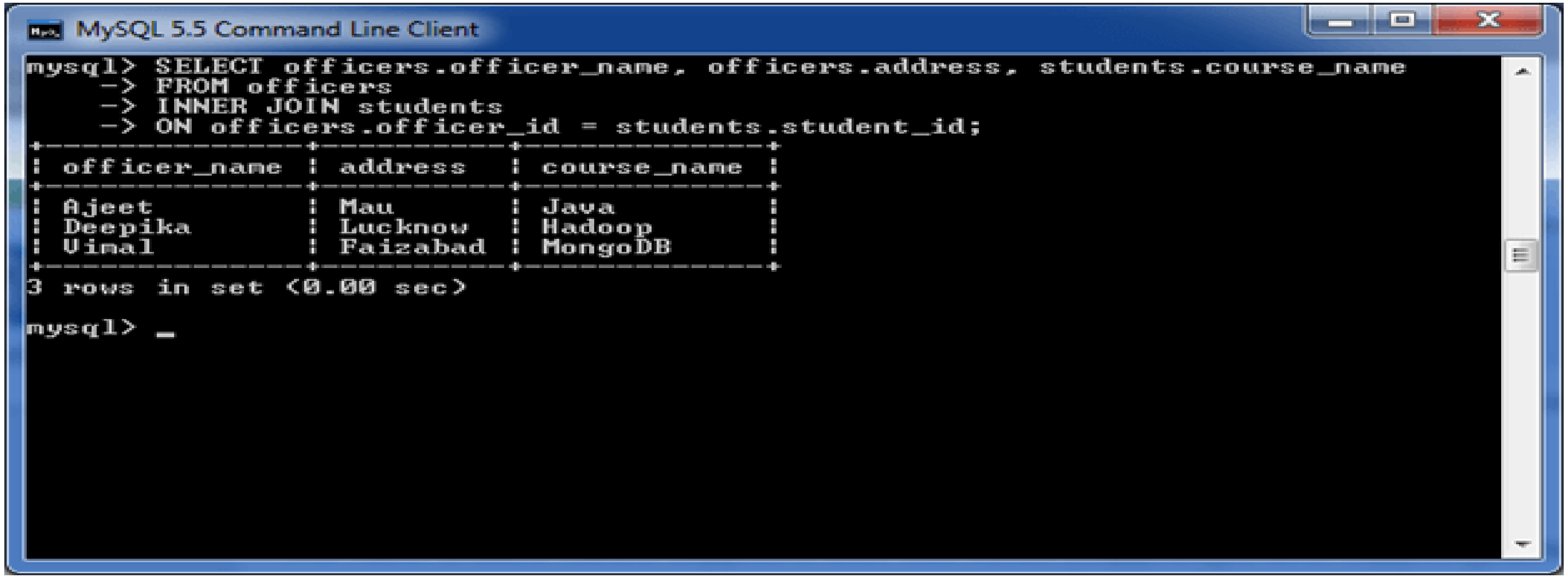
SELECT officers.officer_name, officers.address, students.course_name

FROM officers

INNER JOIN students

ON officers.officer_id = students.student_id;

Output:



```
mysql> SELECT officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> INNER JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+
| officer_name | address  | course_name |
+--------------+----------+-------------+
| Ajeet        | Mau      | Java        |
| Deepika      | Lucknow  | Hadoop      |
| Vimal        | Faizabad | MongoDB     |
+--------------+----------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

2) MySQL Left Outer Join

The LEFT OUTER JOIN returns all rows from the left hand table specified in the ON condition and only those rows from the other table where the join condition is fulfilled.

Syntax:

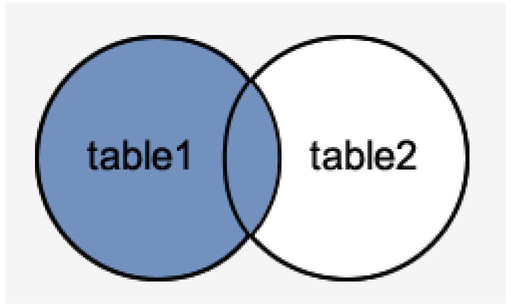
SELECT columns

FROM table1

LEFT [OUTER] JOIN table2

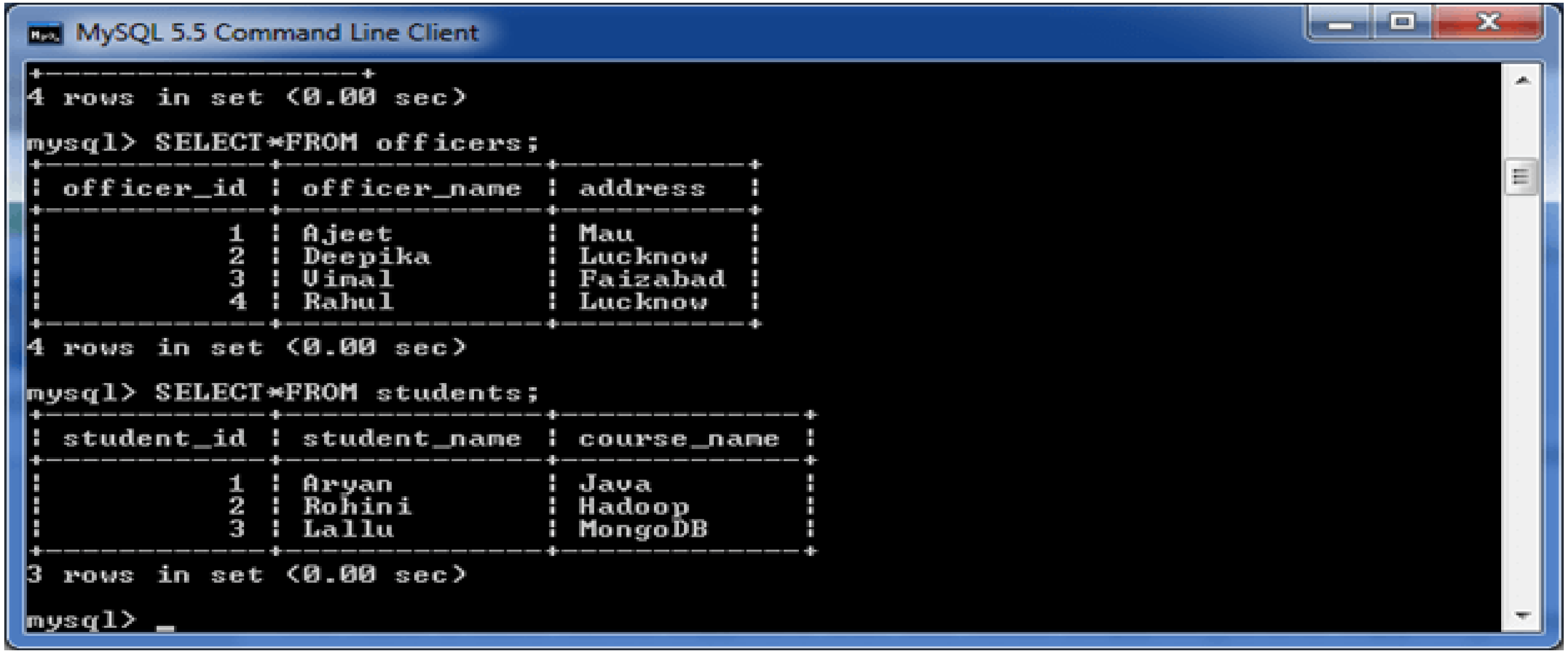
ON table1.column = table2.column;

Image Representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.



```
+------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+-----------+
| officer_id | officer_name | address   |
+------------+--------------+-----------+
|          1 | Ajeet        | Mau       |
|          2 | Deepika      | Lucknow   |
|          3 | Vimal        | Faizabad  |
|          4 | Rahul        | Lucknow   |
+------------+--------------+-----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

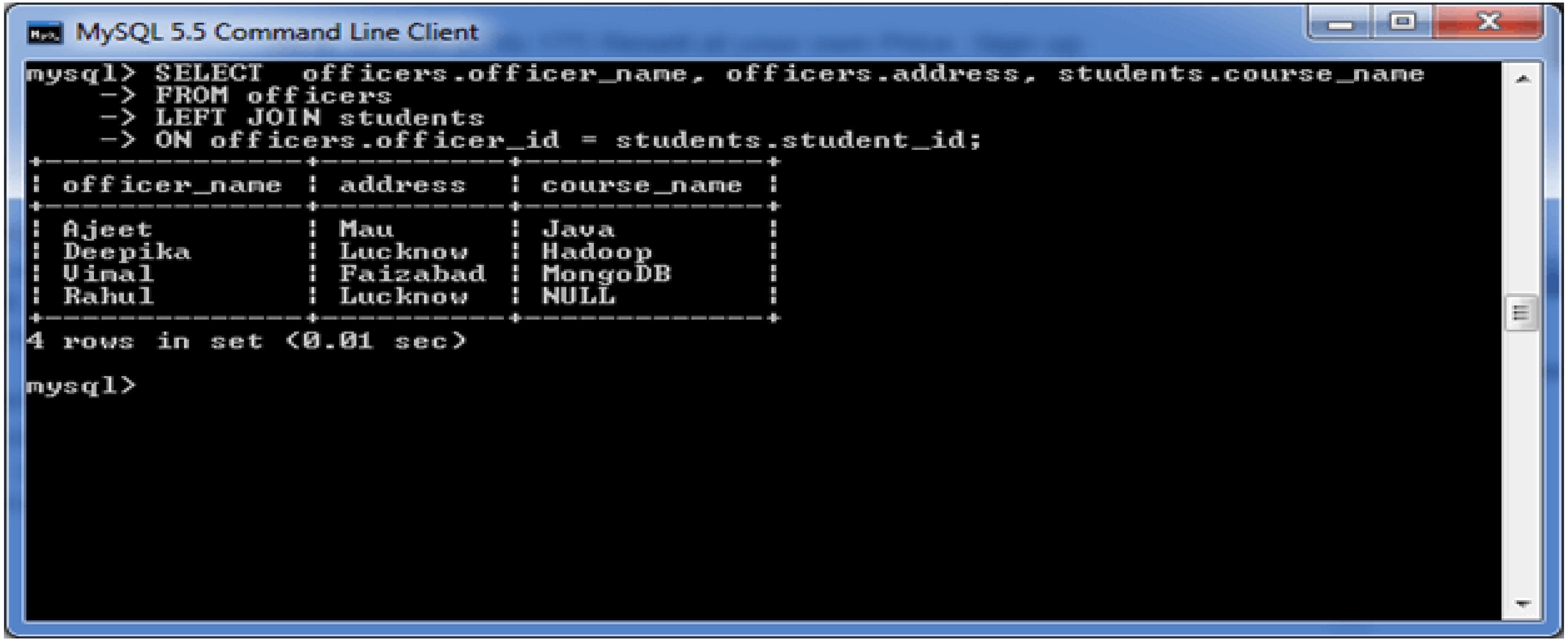
SELECT officers.officer_name, officers.address, students.course_name

FROM officers

LEFT JOIN students

ON officers.officer_id = students.student_id;

Output:



```
mysql> SELECT  officers.officer_name, officers.address, students.course_name
    -> FROM officers
    -> LEFT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+-----------+-------------+
| officer_name | address   | course_name |
+--------------+-----------+-------------+
| Ajeet        | Mau       | Java        |
| Deepika      | Lucknow   | Hadoop      |
| Vimal        | Faizabad  | MongoDB     |
| Rahul        | Lucknow   | NULL        |
+--------------+-----------+-------------+
4 rows in set (0.01 sec)

mysql>
```

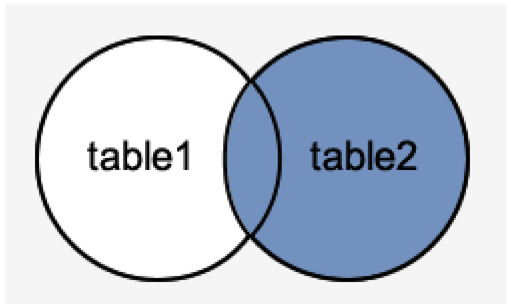
3) MySQL Right Outer Join

The MySQL Right Outer Join returns all rows from the RIGHT-hand table specified in the ON condition and only those rows from the other table where he join condition is fulfilled.

Syntax:

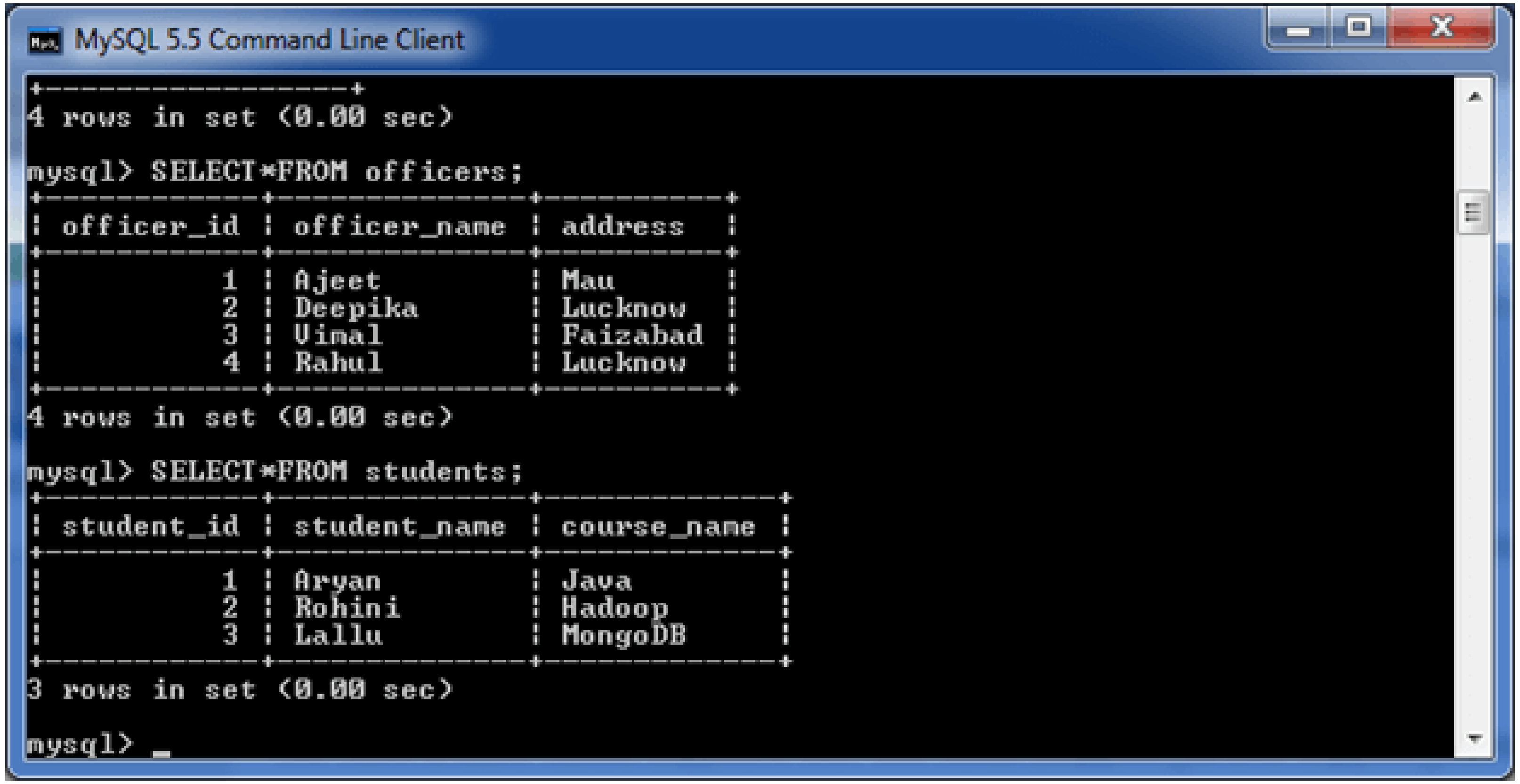
```
SELECT columns
FROM table1
RIGHT [OUTER] JOIN table2
ON table1.column = table2.column;
```

Image representation:



Let's take an example:

Consider two tables "officers" and "students", having the following data.



```
+------------------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM officers;
+------------+--------------+----------+
| officer_id | officer_name | address  |
+------------+--------------+----------+
|          1 | Ajeet        | Mau      |
|          2 | Deepika      | Lucknow  |
|          3 | Vimal        | Faizabad |
|          4 | Rahul        | Lucknow  |
+------------+--------------+----------+
4 rows in set (0.00 sec)

mysql> SELECT*FROM students;
+------------+--------------+-------------+
| student_id | student_name | course_name |
+------------+--------------+-------------+
|          1 | Aryan        | Java        |
|          2 | Rohini       | Hadoop      |
|          3 | Lallu        | MongoDB     |
+------------+--------------+-------------+
3 rows in set (0.00 sec)

mysql> _
```

Execute the following query:

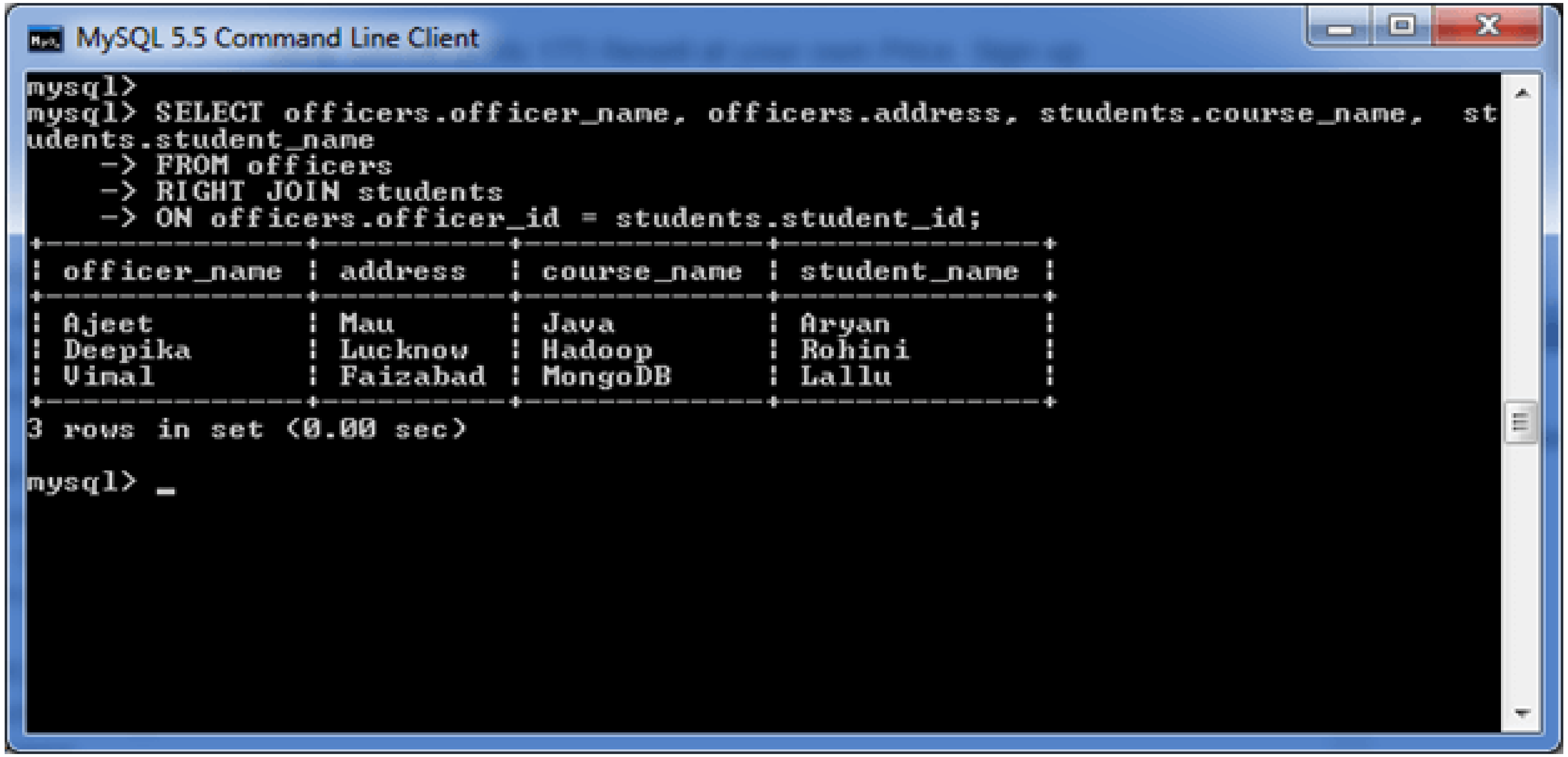
SELECT officers.officer_name, officers.address, students.course_name, students.student_name

FROM officers

RIGHT JOIN students

ON officers.officer_id = students.student_id;

Output:



```
mysql>
mysql> SELECT officers.officer_name, officers.address, students.course_name,  st
udents.student_name
    -> FROM officers
    -> RIGHT JOIN students
    -> ON officers.officer_id = students.student_id;
+--------------+----------+-------------+--------------+
| officer_name | address  | course_name | student_name |
+--------------+----------+-------------+--------------+
| Ajeet        | Mau      | Java        | Aryan        |
| Deepika      | Lucknow  | Hadoop      | Rohini       |
| Vimal        | Faizabad | MongoDB     | Lallu        |
+--------------+----------+-------------+--------------+
3 rows in set (0.00 sec)

mysql> _
```

Views:

A view is a database object that has no values. Its contents are based on the base table. It contains rows and columns similar to the real table. In MySQL, the View is a virtual table created by a query by joining one or more tables. It is operated similarly to the base table but does not contain any data of its own. The View and table have one main difference that the views are definitions built on top of other tables (or views). If any changes occur in the underlying table, the same changes reflected in the View also.

MySQL allows us to create a view in mainly two ways:

- MySQL Command line client
- MySQL Workbench

MySQL Command Line Client

We can create a new view by using the CREATE VIEW and SELECT statement. SELECT statements are used to take data from the source table to make a VIEW.

Syntax

Following is the syntax to create a view in MySQL:

```
CREATE [OR REPLACE] VIEW view_name AS
SELECT columns
FROM tables
[WHERE conditions];
```

Parameters:

The view syntax contains the following parameters:

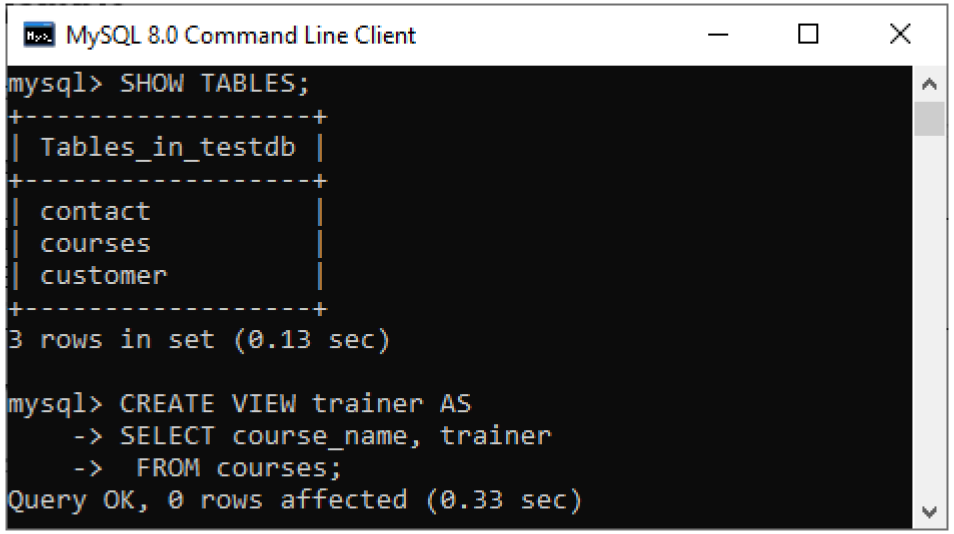
- OR REPLACE: It is optional. It is used when a VIEW already exists. If you do not specify this clause and the VIEW already exists, the CREATE VIEW statement will return an error.
- view_name: It specifies the name of the VIEW that you want to create in MySQL.
- WHERE conditions: It is also optional. It specifies the conditions that must be met for the records to be included in the VIEW.

Example

Let us understand it with the help of an example. Suppose our database has a table course, and we are going to create a view based on this table. Thus, the below example will create a VIEW name "trainer" that creates a virtual table made by taking data from the table courses.

```
CREATE VIEW trainer AS
SELECT course_name, trainer
 FROM courses;
```

Once the execution of the CREATE VIEW statement becomes successful, MySQL will create a view and stores it in the database.


MySQL 8.0 Command Line Client

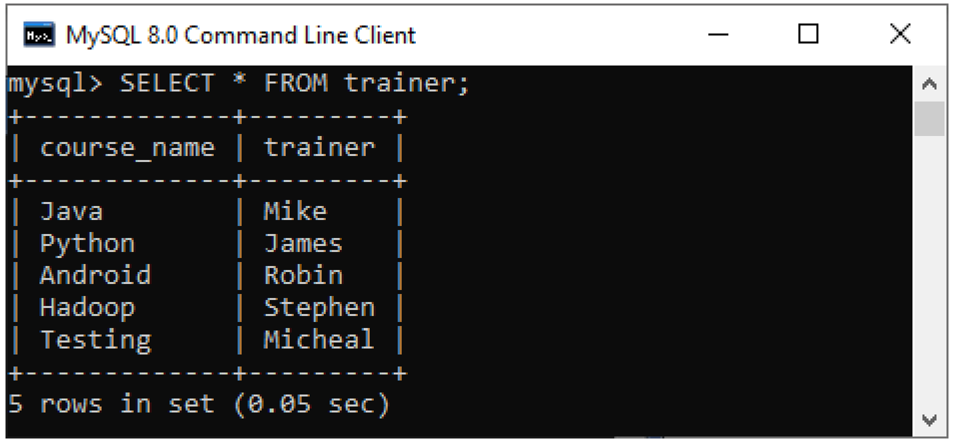
```
mysql> SHOW TABLES;
+------------------+
| Tables_in_testdb |
+------------------+
| contact          |
| courses          |
| customer         |
+------------------+
3 rows in set (0.13 sec)

mysql> CREATE VIEW trainer AS
    -> SELECT course_name, trainer
    ->  FROM courses;
Query OK, 0 rows affected (0.33 sec)
```

To see the created VIEW

We can see the created view by using the following syntax:

- SELECT * FROM view_name;
- Let's see how it looks the created VIEW:
- SELECT * FROM trainer;


MySQL 8.0 Command Line Client

```
mysql> SELECT * FROM trainer;
+-------------+---------+
| course_name | trainer |
+-------------+---------+
| Java        | Mike    |
| Python      | James   |
| Android     | Robin   |
| Hadoop      | Stephen |
| Testing     | Micheal |
+-------------+---------+
5 rows in set (0.05 sec)
```

MySQL Update VIEW

In MYSQL, the ALTER VIEW statement is used to modify or update the already created VIEW without dropping it.

Syntax:

Following is the syntax used to update the existing view in MySQL:

```
ALTER VIEW view_name AS

SELECT columns

FROM table

WHERE conditions;
```

Example:

The following example will alter the already created VIEW name "trainer" by adding a new column.

```
ALTER VIEW trainer AS

SELECT id, course_name, trainer

FROM courses;
```

MySQL Drop VIEW

We can drop the existing VIEW by using the DROP VIEW statement.

Syntax:

The following is the syntax used to delete the view:

DROP VIEW [IF EXISTS] view_name;

Parameters:

- view_name: It specifies the name of the VIEW that we want to drop.
- IF EXISTS: It is optional. If we do not specify this clause and the VIEW doesn't exist, the DROP VIEW statement will return an error.

Example:

Suppose we want to delete the view "trainer" that we have created above. Execute the below statement:

DROP VIEW trainer;

Create View using MySQL Workbench

To create a view in the database using this tool, we first need to launch the MySQL Workbench and log in with the username and password to the MySQL server. It will show the following screen:

1. Go to the Navigation tab and click on the Schema menu. Here, we can see all the previously created databases. Select any database under the Schema menu, for example, testdb. It will pop up the option that can be shown in the following screen.
2. Next, we need to right-click on the view option, and a new pop up screen will come:
3. As soon as we select the "Create View" option, it will give the below screen where we can write our own view.
4. After completing the script's writing, click on the Apply button, we will see the following screen:
5. In that screen, we will review the script and click the Apply button on the database
6. Finally, click on the Finish button to complete the view creation.

MongoDB database:

MongoDB is a popular, open-source NoSQL database management system designed for scalability, flexibility, and performance in handling large volumes of data. Here's a clear explanation of MongoDB:

1. **NoSQL Database:** MongoDB is classified as a NoSQL database, meaning it doesn't adhere to the traditional relational database model (like SQL databases such as MySQL or PostgreSQL). Instead, it provides a more flexible approach to storing and managing data, particularly when dealing with unstructured or semi-structured data.
2. **Document-Oriented:** MongoDB stores data in flexible, JSON-like documents, called BSON (Binary JSON), rather than using tables with rows and columns as in relational databases. Each document can have its own structure, allowing for dynamic schemas where different documents in the same collection (similar to tables in SQL) can have different fields.
3. **Scalability:** MongoDB is designed to scale horizontally across multiple servers, making it suitable for handling large amounts of data and high-traffic applications. It supports sharding, which involves distributing data across multiple machines to improve performance and accommodate growth.

4. **High Performance:** MongoDB offers high performance for both read and write operations. It achieves this through various mechanisms such as indexing, which allows for efficient querying, and in-memory storage engine optimizations.

5. **Querying and Indexing:** MongoDB provides a rich query language with support for a wide range of operations such as filtering, sorting, aggregation, and text search. It also supports indexing, which can significantly improve query performance by allowing the database to quickly locate relevant data.

6. **Replication and Fault Tolerance:** MongoDB supports replica sets, which are groups of MongoDB servers that maintain the same data set for fault tolerance and high availability. In a replica set, one node serves as the primary, handling all write operations, while the others act as secondaries, replicating data from the primary and serving read operations.

7. **Flexibility:** MongoDB's schema-less design allows for greater flexibility in data modeling compared to relational databases. Developers can easily evolve the data model as application requirements change without needing to modify existing data or schema definitions.

8. **Use Cases:** MongoDB is well-suited for a variety of use cases, including content management systems, real-time analytics, mobile app development, IoT (Internet of Things) applications, and more. Its flexibility, scalability, and performance make it a popular choice for modern, data-intensive applications.

4.Importing-exporting and querying data

Importing, exporting, and querying data are fundamental operations in working with databases and data management systems. Here's a clear explanation of each:

Importing Data:

Importing data involves bringing data from an external source into a database or data management system. This can be done through various means such as:

- File Upload: Uploading data files (like CSV, Excel, JSON, etc.) directly into the system.
- Data Integration Tools: Using specialized software or tools to connect to external data sources (like APIs, other databases, web services) and pull data into the system.
- Database Migration: Transferring data from one database system to another.
- Data Entry: Manually entering data into the system.

The goal of importing data is to populate the database with relevant information for analysis, storage, or processing.

Exporting Data:

Exporting data involves retrieving data from a database or data management system and transferring it to an external destination. This can include:

- File Download: Downloading data in various formats (CSV, Excel, JSON, etc.) from the system to local storage.
- Integration with Other Systems: Sending data to other systems for further processing or analysis.
- Reporting: Generating reports based on database contents and exporting them for sharing or archival purposes.

Exporting data is essential for sharing information, generating reports, or integrating with other applications.

Querying Data:

Querying data involves retrieving specific information from a database based on certain criteria or conditions. This is typically done using a query language such as SQL (Structured Query Language) for relational databases. Here's how it works:

- SELECT Statement: Used to retrieve data from one or more tables in the database.
- FILTERing: Adding conditions to the query to retrieve only the data that meets certain criteria.
- JOINs: Combining data from multiple tables based on common keys.
- Aggregate Functions: Performing calculations on data, such as SUM, AVG, COUNT, etc.

The result of a query is usually a subset of the database that meets the specified conditions.

5. creating and manipulating documents

Creating and manipulating documents in MySQL typically involves working with JSON data types. MySQL supports JSON as a native data type since version 5.7, allowing you to store, index, and query JSON documents directly within the database.

1. **Creating a Table with JSON Data Type:** To start, you need a table with a column of type JSON where you'll store your documents. You can create such a table like this:

```
CREATE TABLE documents (
id INT AUTO_INCREMENT PRIMARY KEY,
data JSON
);
```

This creates a table named documents with two columns: id as the primary key and data to store JSON documents.

2. **Inserting JSON Documents:**

Once you have the table, you can insert JSON documents into it. For example:

```
INSERT INTO documents (data) VALUES
    ('{"name": "John", "age": 30, "city": "New York"}'),
    ('{"name": "Alice", "age": 25, "city": "Los Angeles"}');
```

3. Querying JSON Data: MySQL provides various functions to query JSON data. For instance, you can use JSON_EXTRACT() to retrieve specific fields from JSON documents:

```
SELECT JSON_EXTRACT(data, '$.name') AS name FROM documents;
```

This query extracts the name field from each JSON document stored in the data column.

4. **Updating JSON Documents:**

You can update JSON documents using the JSON_SET() function:

```
UPDATE documents

SET data = JSON_SET(data, '$.age', 31)

WHERE id = 1;
```

This query updates the age field of the JSON document with id equal to 1 to 31.

5. **Deleting JSON Documents:** To delete JSON documents, you can use standard DELETE queries:

```
DELETE FROM documents

 WHERE id = 2;
```

This deletes the JSON document with id equal to 2 from the documents table.

6. Manipulating Nested JSON Data:

MySQL also provides functions to manipulate nested JSON data. For example, you can use JSON_OBJECT() to construct JSON objects dynamically:

```
SELECT JSON_OBJECT('name', 'Bob', 'age', 28) AS new_data;
```

This constructs a new JSON object with the name and age fields.

6. CRUD operation

CRUD stands for Create, Read, Update, and Delete, which are the four basic functions of persistent storage. In MySQL, you can perform CRUD operations using SQL queries. Here's a brief overview of how to perform CRUD operations in MySQL:

1. Create: To insert new data into a table, you use the INSERT INTO statement.

```
INSERT INTO table_name (column1, column2, ...)

VALUES (value1, value2, ...);
```

Example:

```
INSERT INTO users (name, email)

VALUES ('John Doe', 'john@example.com’);
```

2. Read: To retrieve data from a table, you use the SELECT statement.

```
SELECT column1, column2, ...

FROM table_name

WHERE condition;
```

Example:

```
SELECT * FROM users;
```

3. Update: To modify existing data in a table, you use the UPDATE statement.

```
UPDATE table_name
SET column1 = value1, column2 = value2, ...
WHERE condition;
```

Example:

```
UPDATE users
SET email = 'newemail@example.com'
WHERE id = 1;
```

4. Delete: To remove data from a table, you use the DELETE FROM statement.

```
DELETE FROM table_name
WHERE condition;
```

Example:

```
DELETE FROM users
WHERE id = 1;
```

7. Indexing and aggregation pipeline

Indexing and aggregation are crucial techniques for optimizing query performance and performing complex data analysis, but they work differently compared to MongoDB. Let's break down both concepts:

1. Indexing in MySQL:

Indexing in MySQL involves creating data structures that help accelerate the retrieval of data from tables. Indexes are created on columns or expressions within tables and enable the database engine to quickly locate rows that match specified criteria in SELECT, UPDATE, DELETE, and JOIN operations.

Types of Indexes in MySQL:

- **Primary Key Index:** Automatically created when defining a PRIMARY KEY constraint on a column or a combination of columns. It uniquely identifies each row in a table.
- **Unique Index:** Ensures that the values in a column or a combination of columns are unique across the table.
- **Index:** A regular index created explicitly on one or more columns to speed up data retrieval.
- **Full-Text Index:** Special index type optimized for full-text searches on text columns.

- **Composite Index:** Index created on multiple columns to improve performance for queries involving those columns.
- **Spatial Index:** Used for spatial data types to optimize spatial queries.
- **Hash Index:** Introduced in MySQL 8.0 for memory-optimized tables, hash indexes provide efficient equality lookups.

Aggregation in MySQL:

In MySQL, aggregation involves summarizing and analyzing data using aggregate functions such as COUNT, SUM, AVG, MIN, MAX, etc. Aggregation operations are typically performed with the GROUP BY clause, which groups rows that have the same values into summary rows, and then applies aggregate functions to those groups.

Example Aggregation Query in MySQL:

```
SELECT
    department_id,
    COUNT(*) AS employee_count,
    AVG(salary) AS average_salary
FROM
    employees
GROUP BY
    department_id;
```

In this example:

- We're calculating the number of employees (employee_count) and the average salary (average_salary) for each department.
- The GROUP BY clause groups the rows by the department_id.
- Aggregation operations allow for data summarization, statistical analysis, and reporting.

Comparison:

While indexing and aggregation serve similar purposes in both MongoDB and MySQL, the implementations differ:

- In MySQL, indexes optimize data retrieval by creating data structures associated with tables, while in MongoDB, indexes are a part of the document storage.
- Aggregation in MySQL involves using aggregate functions and the GROUP BY clause, whereas MongoDB utilizes the aggregation pipeline framework for complex data transformations.