Agenda:

- How to create and drop Database ?
- What is a table in a Database ?
- Data types in mysql.
- How to create and drop tables ?
- What are primary and unique keys ?
- Insert, Update, Delete and Select statements.
- Conditionals and Operators.

In the last session, we have learnt about the basics of Databases and Database Management Systems like mySQL etc.

In this session, we are going to start learning about the mySQL, for this we have installed XAMPP on our computers in the last session. And hence, w have installed mySQL on our computers through XAMPP.

So, in this session we are going to start with the CRUD operations on mySQL using both command line of mySQL which we have installed throug XAMPP and also through GUI(Graphical User Interface) using XAMPP.

Getting started with mySQL through command line:

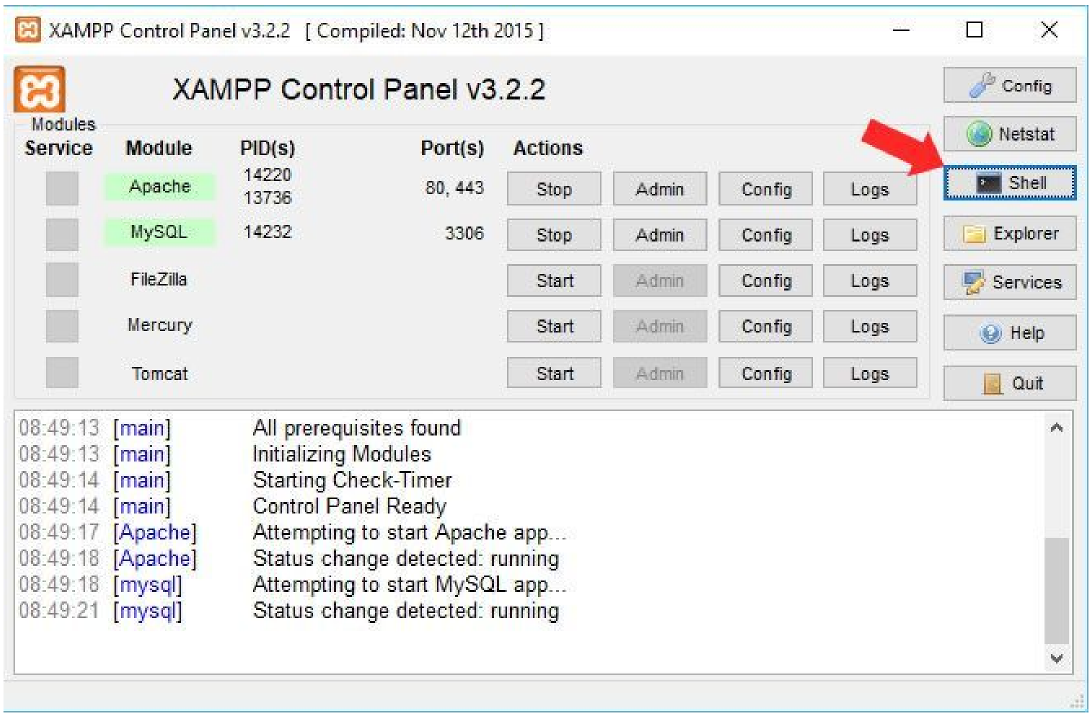
Opening mySQL server on command line:

In order to open the command line for your mySQL server,

Firstly, you have to go to the XAMPP folder on your computer, then under this folder open mySQL folder then under this folder you will find the bin fold this is the folder where all the mySQL servers are present on your computer.

Now, open the command line on this folder.

Or you can open the command line for mySQL server through XAMPP as shown below:



```
08:49:13  [main]     All prerequisites found
08:49:13  [main]     Initializing Modules
08:49:14  [main]     Starting Check-Timer
08:49:14  [main]     Control Panel Ready
08:49:17  [Apache]   Attempting to start Apache app...
08:49:18  [Apache]   Status change detected: running
08:49:18  [mysql]    Attempting to start MySQL app...
08:49:21  [mysql]    Status change detected: running
```

- click XAMPP icon to launch its cPanel
- click on the shell button

This will open the command line for mySQL server.

Connecting to mySQL server:

After opening the command line for mySQL server, run the following command on your shell:

```
mysql -h 127.0.0.1 -u root
```

After this command you would see the following details and you will be ready to do operations with mysql server:

```
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.6.21 MySQL Community Server (GPL)

Copyright (c) 2000, 2014, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

Now, we are ready to create our database in mysql.

Creating a Database:

To create a new database in MySQL, you use the `CREATE DATABASE` statement with the following syntax:

```
CREATE DATABASE [IF NOT EXISTS] database_name
[CHARACTER SET charset_name]
[COLLATE collation_name]
```

In this syntax:

- First, specify name of the database after the the `CREATE DATABASE` keywords. The database name must be unique within a MySQL server instance. If you attempt to create a database with a name that already exists, MySQL will issue an error.
- Second, use the `IF NOT EXISTS` option to conditionally create a database if it doesn't exist.
- Third, specify the character set and collation for the new database. If you skip the `CHARACTER SET` and `COLLATE` clauses, MySQL will the default character set and collation for the new database.

Next, we can check the current databases available on the server using the `SHOW DATABASE` statement. This step is optional.

```
SHOW DATABASES;
```

Droping a Database:

The `DROP DATABASE` statement drops all tables in the database and deletes the database permanently. Therefore, you need to be very careful whe using this statement.

The following shows the syntax of the `DROP DATABASE` statement:

```
DROP DATABASE [IF EXISTS] database_name;
```

In this statement, you specify the name of the database which you want to delete after the `DROP DATABASE` keywords.

If you drop a database that does not exist, MySQL will issue an error.

To prevent an error from occurring if you delete a non-existing database, you can use the `IF EXISTS` option. In this case, MySQL will terminate tl statement without issuing any error.

The `DROP DATABASE` statement returns the number of tables it deleted.

In MySQL, the schema is the synonym for the database. Therefore, you can use them interchangeably:

```
DROP SCHEMA [IF EXISTS] database_name;
```

Tables in mySQL:

A table is used to organize data in the form of rows and columns and used for both storing and displaying records in the structure format. It is similar worksheets in the spreadsheet application.

Data types in mySQL:

A database table contains multiple columns with specific data types such as numeric or string. MySQL provides more data types other than just numer and string. Each data type in MySQL can be determined by the following characteristics:

- The kind of values it represents.
- The space that takes up and whether the values are a fixed-length or variable length.
- The values of the data type can be indexed or not.
- How MySQL compares the values of a specific data type.

INT data type:

In MySQL, `INT` stands for the integer that is a whole number. An integer can be written without a fractional component e.g., 1, 100, 4, -10, and it cann be 1.2, 5/3, etc. An integer can be zero, positive, and negative.

MySQL supports all standard SQL integer types `INTEGER` or `INT` and `SMALLINT` . In addition, MySQL provides `TINYINT` `MEDIUMINT` and `BIGINT` as extensions to the SQL standard.

MySQL `INT` data type can be signed and unsigned.

BIT data type:

The `BIT` type that allows you to store bit values. Here is the syntax:

```
BIT(n)
```

The `BIT(n)` can store up to n-bit values. The `n` can range from 1 to 64.

CHAR data type:

The `CHAR` data type is a fixed-length character type in MySQL. You often declare the `CHAR` type with a length that specifies the maximum number characters that you want to store. For example, `CHAR(20)` can hold up to 20 characters.

The length of the `CHAR` data type can be any value from 0 to 255. When you store a `CHAR` value, MySQL pads its value with spaces to the length th you declared.

BOOLEAN data type:

MySQL does not have built-in Boolean type. However, it uses `[TINYINT(1)](https://www.mysqltutorial.org/mysql-int/)` instead. To make more convenient, MySQL provides `BOOLEAN` or `BOOL` as the synonym of `TINYINT(1)` .

In MySQL, zero is considered as false, and non-zero value is considered as true. To use Boolean literals, you use tl constants `TRUE` and `FALSE` that evaluate to 1 and 0 respectively. See the following example:

```
SELECT true, false, TRUE, FALSE, True, False;
-- 1 0 1 0 1 0
```

So, these are the basic data types in MySQL, and if you want to learn about other data types you can visit the followir link:https://blog.devart.com/mysql-data-types.html

Creating a Table:

The `CREATE TABLE` statement allows you to create a new table in a database.

The following illustrates the basic syntax of the `CREATE TABLE` statement:

```
CREATE TABLE [IF NOT EXISTS] table_name(
   column_1_definition,
   column_2_definition,
   ...,
   table_constraints
)
```

Let’s examine the syntax in greater detail.

First, you specify the name of the table that you want to create after the `CREATE TABLE` keywords. The table name must be unique within a databas The `IF NOT EXISTS` is optional. It allows you to check if the table that you create already exists in the database. If this is the case, MySQL will igno the whole statement and will not create any new table.

Second*,* you specify a list of columns of the table in the `column_list` section, columns are separated by commas.

As we have seen in above syntax, there are columns definitions we need to define while creating a table, so let's see how we can define the columns in table:

The following shows the syntax for a column's definition:

```
column_name data_type(length) [NOT NULL] [DEFAULT value] [AUTO_INCREMENT] column_constraint;
```

Here are the details:

- The `column_name` specifies the name of the column. Each column has a specific data type and optional size e.g., `VARCHAR(255)`
- The `NOT NULL` constraint ensures that the column will not contain `NULL` . Besides the `NOT NULL` constraint, a column may have additional constraint such as CHECK, and UNIQUE.
- The `DEFAULT` specifies a default value for the column.
- The `AUTO_INCREMENT` indicates that the value of the column is incremented by one automatically whenever a new row is inserted into the table. Each table has a maximum one `AUTO_INCREMENT` column.

Now, at the end of the column definition we can see the word column_constraint, so,

After the column list, you can define table constraints such as UNIQUE, CHECK, PRIMARY KEY and FOREIGN KEY.

For example, if you want to set a column or a group of columns as the primary key, you use the following syntax:

```
PRIMARY KEY (col1,col2,...)
```

NOTE: We will learn about these keys and constraints later in detail.

Now, lets see the concept of creating table using an example:

The following statement creates a new table named `tasks` :

```
CREATE TABLE IF NOT EXISTS tasks (
    task_id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    start_date DATE,
    due_date DATE,
    status TINYINT NOT NULL,
    priority TINYINT NOT NULL,
    description TEXT,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
)
```

The tasks table has the following columns:

- The `task_id` is an auto-increment column. If you use the `INSERT` statement to insert a new row into the table without specifying a value for the `task_id` column, MySQL will automatically generate a sequential integer for the `task_id` starting from 1.
- The `title` column is a variable character string column whose maximum length is 255. It means that you cannot insert a string whose length is greater than 255 into this column. The `NOT NULL` constraint indicates that the column does not accept `NULL` . In other words, you have to provide a non-NULL value when you insert or update this column.
- The `start_date` and `due_date` are `DATE` columns. Because these columns do not have the `NOT NULL` constraint, they can store `NULL` . The start_date column has a default value of the current date. In other words, if you don't provide a value for the start_date column when you insert a new row, the start_date column will take the current date of the database server.
- The `status` and `priority` are the `TINYINT` columns which do not allow `NULL` .
- The `description` column is a `TEXT` column that accepts `NULL` .
- The `created_at` is a `TIMESTAMP` column that accepts the current time as the default value.

The `task_id` is the primary key column of the `tasks` table. It means that the values in the `task_id` column will uniquely identify rows in the table

Droping a Table:

To remove existing tables, you use the MySQL `DROP TABLE` statement.

Here is the basic syntax of the `DROP TABLE` statement:

```
DROP [TEMPORARY] TABLE [IF EXISTS] table_name [, table_name] ...
[RESTRICT | CASCADE]
```

The `DROP TABLE` statement removes a table and its data permanently from the database. In MySQL, you can also remove multiple tables using single `DROP TABLE` statement, each table is separated by a comma (,).

The `TEMPORARY` option allows you to remove temporary tables only. It ensures that you do not accidentally remove non-temporary tables.

The `IF EXISTS` option conditionally drop a table only if it exists. If you drop a non-existing table with the `IF EXISTS` option, MySQL generates NOTE, which can be retrieved using the `SHOW WARNINGS` statement.

Note that the `DROP TABLE` statement only drops tables. It doesn't remove specific user privileges associated with the tables. Therefore, if you create table with the same name as the dropped one, MySQL will apply the existing privileges to the new table, which may pose a security risk.

The `RESTRICT` and `CASCADE` options are reserved for the future versions of MySQL.

To execute the `DROP TABLE` statement, you must have `DROP` privileges for the table that you want to remove.

Now, after learning how to create and drop tables, let's start learning about the keys :

Keys in mySQL:

A key as the term itself indicates, **unlocks access to the tables**. If you know the key, you know how to identify specific records and the relationshi between the tables. Each key consists of one or more fields, or field prefix. The order of columns in an index is significant.

Primary Keys:

A primary key is a column or a set of columns that uniquely identifies each row in the table. The primary key follows these rules:

- A primary key must contain unique values. If the primary key consists of multiple columns, the combination of values in these columns must be unique.
- A primary key column cannot have `NULL` values. Any attempt to insert or update `NULL` to primary key columns will result in an error. Note that MySQL implicitly adds a `NOT NULL` constraint to primary key columns.
- A table can have one an only one primary key.

A primary key column often has the `AUTO_INCREMENT` attribute that automatically generates a sequential integer whenever you insert a new row in the table.

When you define a primary key for a table, MySQL automatically creates a new index called `PRIMARY` .

Unique key:

`KEY` is the synonym for `INDEX` . A `UNIQUE` index ensures that values in a column must be unique. Unlike the `PRIMARY` index, MySC allows `NULL` values in the `UNIQUE` index. In addition, a table can have multiple `UNIQUE` indexes.

Suppose that `email` and `username` of users in the `users` table must be unique. To enforce thes rules, you can define `UNIQUE` indexes f the `email` and `username` columns as the following statement:

Add a `UNIQUE` index for the `username` column:

```
ALTER TABLE users
ADD UNIQUE INDEX username_unique (username ASC) ;
```

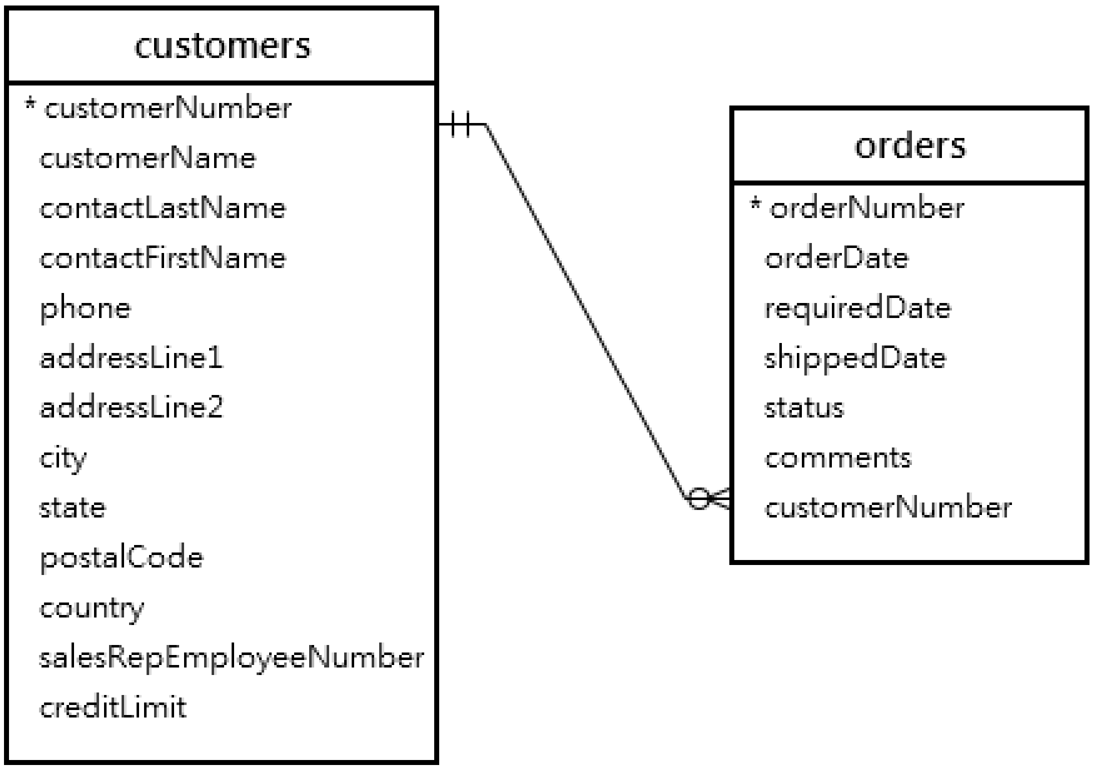
Add a `UNIQUE` index for the `email` column:

```
ALTER TABLE users
ADD UNIQUE INDEX  email_unique (email ASC) ;
```

Foreign Keys:

A foreign key is a column or group of columns in a table that links to a column or group of columns in another table. The foreign key places constraints c data in the related tables, which allows MySQL to maintain referential integrity.

Let's take a look at the following `customers` and `orders` tables.



In this diagram, each customer can have zero or many orders and each order belongs to one customer.

The relationship between `customers` table and `orders` table is one-to-many. And this relationship is established by the foreign key the `orders` table specified by the `customerNumber` column.

The `customerNumber` column in the `orders` table links to the `customerNumber` primary key column in the `customers` table.

The `customers` table is called the *parent table* or *referenced table*, and the `orders` table is known as the *child table* or *referencing table*.

Typically, the foreign key columns of the child table often refer to the primary key columns of the parent table.

A table can have more than one foreign key where each foreign key references to a primary key of the different parent tables.

Once a foreign key constraint is in place, the foreign key columns from the child table must have the corresponding row in the parent key columns of tl parent table or values in these foreign key column must be `NULL` (see the `SET NULL` action example below).

For example, each row in the `orders` table has a `customerNumber` that exists in the `customerNumber` column of the `customers` table. Multip rows in the `orders` table can have the same `customerNumber` .

Now, as we have learnt about the keys, we can start learning operations:

SELECT statement:

The `SELECT` statement allows you to select data from one or more tables. To write a `SELECT` statement in MySQL, you use this syntax:

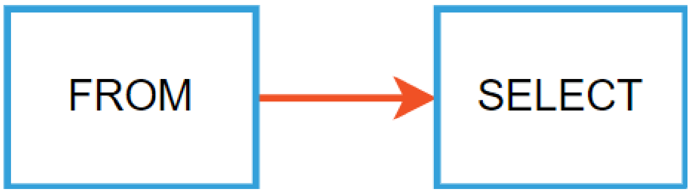
```
SELECT select_list
FROM table_name;
```

In this syntax:

- First, specify one or more columns from which you want to select data after the `SELECT` keyword. If the `select_list` has multiple columns, you need to separate them by a comma (`,`).
- Second, specify the name of the table from which you want to select data after the `FROM` keyword.

The semicolon (`;`) is optional. It denotes the end of a statement. If you have two or more statements, you need to use the semicolon(`;)` to separa them so that MySQL will execute each statement individually.

When executing the `SELECT` statement, MySQL evaluates the `FROM` clause before the `SELECT` clause:



ORDER BY statement:

When you use the `SELECT` statement to query data from a table, the order of rows in the result set is unspecified. To sort the rows in the result set, yo
add the `ORDER BY` clause to the `SELECT` statement.

The following illustrates the syntax of the `ORDER BY` clause:

```
SELECT
   select_list
FROM
   table_name
ORDER BY
   column1 [ASC|DESC],
   column2 [ASC|DESC],
   ...;
```

In this syntax, you specify the one or more columns that you want to sort after the `ORDER BY` clause.

The `ASC` stands for ascending and the `DESC` stands for descending. You use `ASC` to sort the result set in ascending order and `DESC` to sort th
result set in descending order respectively.

INSERT statement:

The `INSERT` statement allows you to insert one or more rows into a table. The following illustrates the syntax of the `INSERT` statement:

```
INSERT INTO table(c1,c2,...)
VALUES (v1,v2,...);
```

In this syntax,

- First, specify the table name and a list of comma-separated columns inside parentheses after the `INSERT INTO` clause.
- Then, put a comma-separated list of values of the corresponding columns inside the parentheses following the `VALUES` keyword.

The number of columns and values must be the same. In addition, the positions of columns must be corresponding with the positions of their values.

To insert multiple rows into a table using a single `INSERT` statement, you use the following syntax:

```
INSERT INTO table(c1,c2,...)
VALUES
   (v11,v12,...),
   (v21,v22,...),
    ...
   (vnn,vn2,...);
```

In this syntax, rows are separated by commas in the `VALUES` clause.

Consider the example shown below:

We have created a tasks table when we are learning how to create tables, so now let's insert a row in that table using INSERT :

The following statement inserts a new row into the `tasks` table:

```
INSERT INTO tasks(title,priority)
VALUES('Learn MySQL INSERT Statement',1);
```

MySQL returns the following message:

```
1 row(s) affected
```

It means that one row has been inserted into the `tasks` table successfully.

UPDATE statement:

The `UPDATE` statement updates data in a table. It allows you to change the values in one or more columns of a single row or multiple rows.

The following illustrates the basic syntax of the `UPDATE` statement:

```
UPDATE [LOW_PRIORITY] [IGNORE] table_name
SET
    column_name1 = expr1,
    column_name2 = expr2,
    ...
[WHERE
    condition];
```

In this syntax:

- First, specify the name of the table that you want to update data after the `UPDATE` keyword.
- Second, specify which column you want to update and the new value in the `SET` clause. To update values in multiple columns, you use a list of comma-separated assignments by supplying a value in each column's assignment in the form of a literal value, an expression, or a subquery.
- Third, specify which rows to be updated using a condition in the `WHERE` clause. The `WHERE` clause is optional. If you omit it, the `UPDATE` statement will modify all rows in the table.

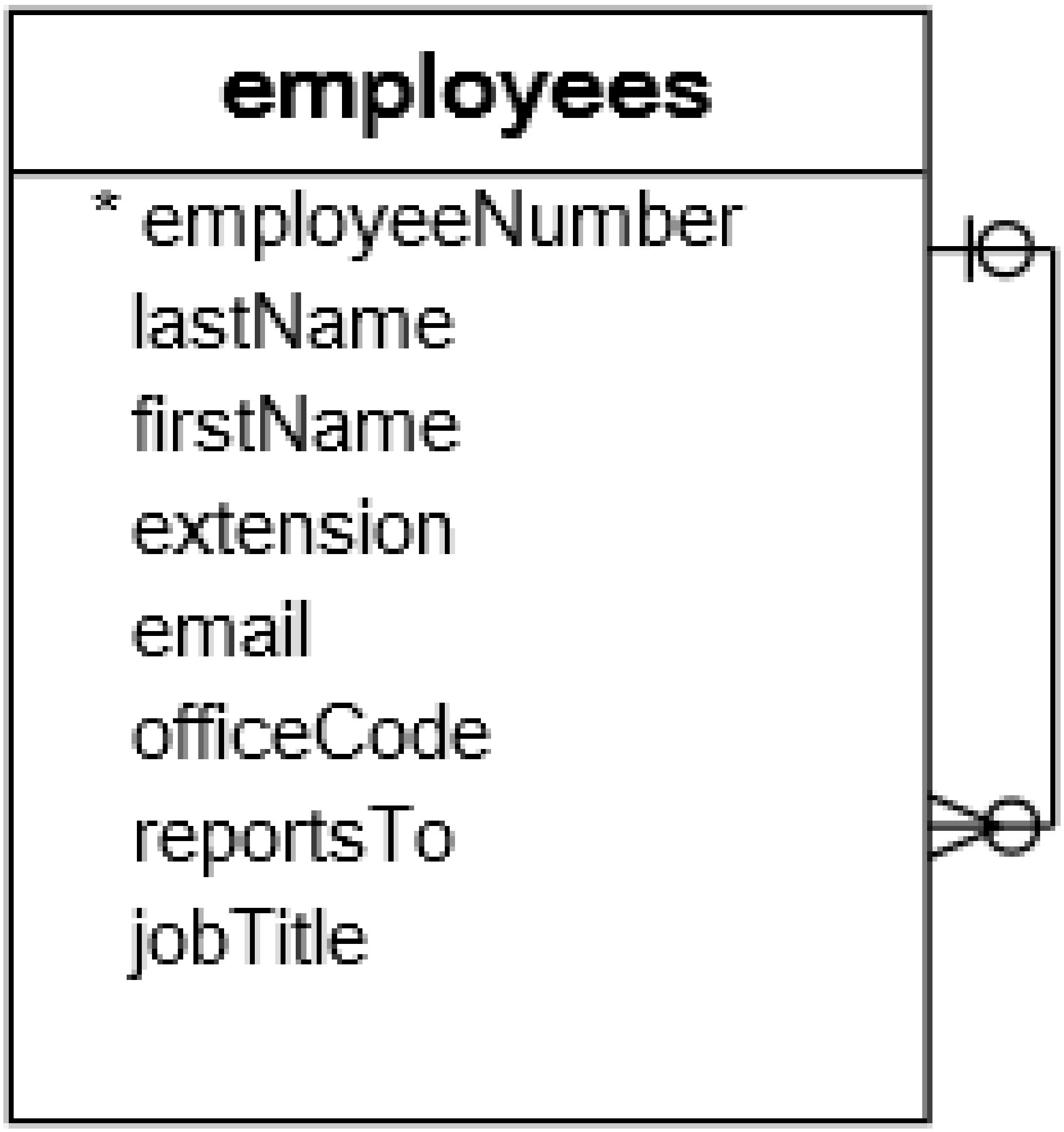
Notice that the `WHERE` clause is so important that you should not forget. Sometimes, you may want to update just one row; However, you may forg the `WHERE` clause and accidentally update all rows of the table.

MySQL supports two modifiers in the `UPDATE` statement.

1. The `LOW_PRIORITY` modifier instructs the `UPDATE` statement to delay the update until there is no connection reading data from the table. The `LOW_PRIORITY` takes effect for the storage engines that use table-level locking only such as `MyISAM` , `MERGE` , and `MEMORY` .
2. The `IGNORE` modifier enables the `UPDATE` statement to continue updating rows even if errors occurred. The rows that cause errors such as duplicate-key conflicts are not updated.

Let's understand this with the example given below:

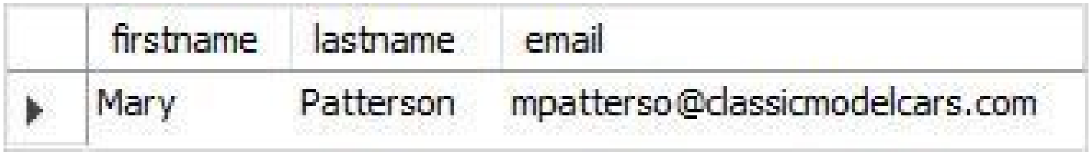
See the following `employees` table.



In this example, we will update the email of `Mary Patterson` to the new email `mary.patterso@classicmodelcars.com` .

First, find Mary's email from the `employees` table using the following `SELECT` statement:

```
SELECT
    firstname,
    lastname,
    email
FROM
    employees
WHERE
    employeeNumber = 1056;
Code language: SQL (Structured Query Language) (sql)
```



	firstname	lastname	email
▶	Mary	Patterson	mpatterso@classicmodelcars.com

Second, update the email address of `Mary` to the new email `mary.patterson@classicmodelcars.com` :

```
UPDATE employees
SET
    email = 'mary.patterson@classicmodelcars.com'
```

```
WHERE
    employeeNumber = 1056;
```

MySQL issued the number of rows affected:

```
1 row(s) affected
```

DELETE statement:

To delete data from a table, you use the MySQL `DELETE` statement. The following illustrates the syntax of the `DELETE` statement:

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

In this statement:

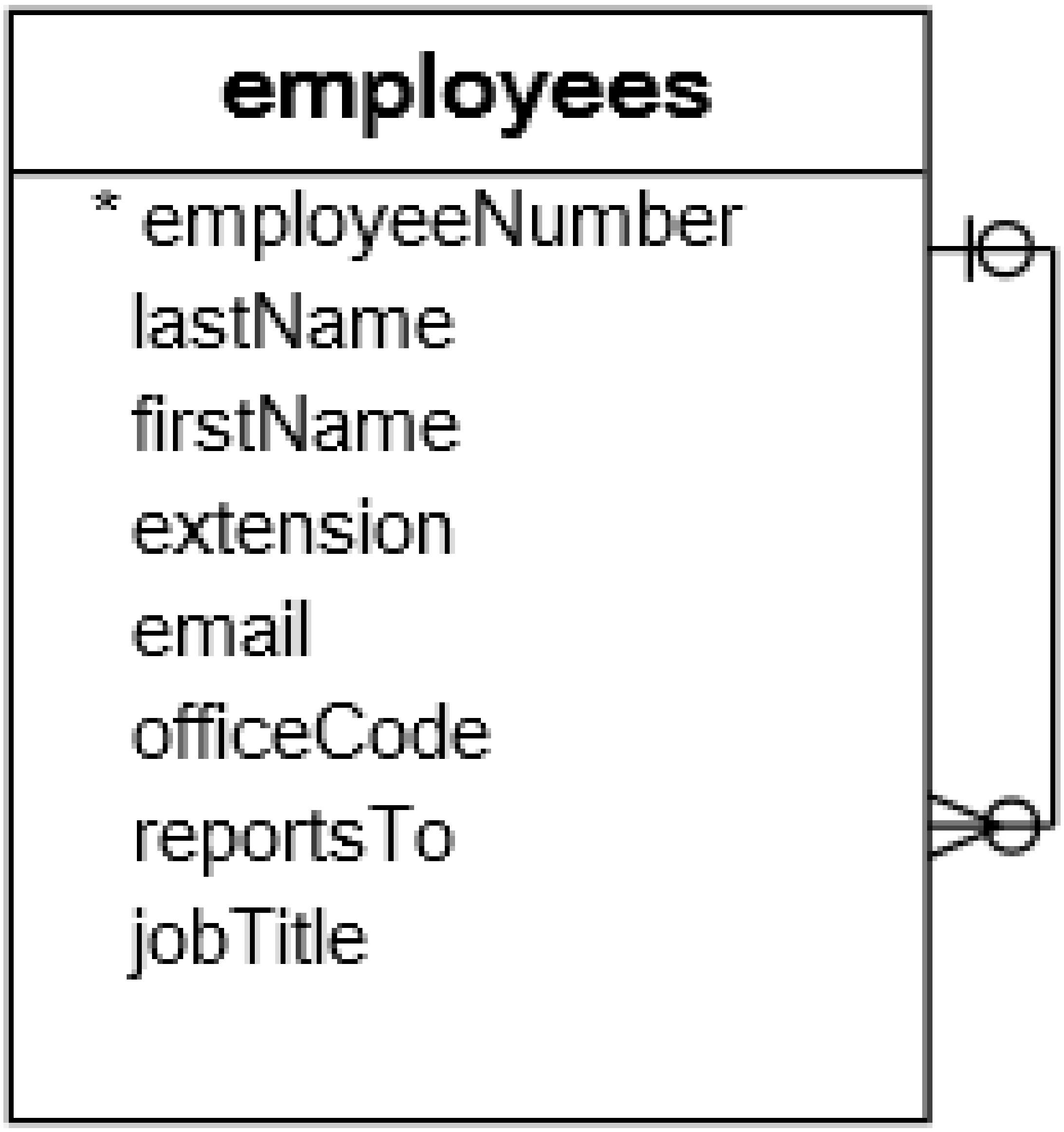
- First, specify the table from which you delete data.
- Second, use a condition to specify which rows to delete in the `WHERE` clause. The `DELETE` statement will delete rows that match the condition,

Notice that the `WHERE` clause is optional. If you omit the `WHERE` clause, the `DELETE` statement will delete all rows in the table.

Besides deleting data from a table, the `DELETE` statement returns the number of deleted rows.

Let's understand this with the example given below:

We will use the `employees` table.



Note that once you delete data, it is gone. Later, you will learn how to put the `DELETE` statement in a transaction so that you can roll it back.

Suppose you want to delete employees whose the `officeNumber` is 4, you use the `DELETE` statement with the `WHERE` clause as shown in tl following query:

```
DELETE FROM employees
WHERE officeCode = 4;
```

To delete all rows from the `employees` table, you use the `DELETE` statement without the `WHERE` clause as follows:

```
DELETE FROM employees;
```

All rows in the `employees` table deleted.

Now, we have learnt about the main statements in mySQL, let's start learning about the conditional statements and main operators:

Conditional Statements and Operators:

WHERE clause:

The `WHERE` clause allows you to specify a search condition for the rows returned by a query. The following shows the syntax of the `WHERE` clause:

```
SELECT
    select_list
FROM
    table_name
WHERE
    search_condition;
```

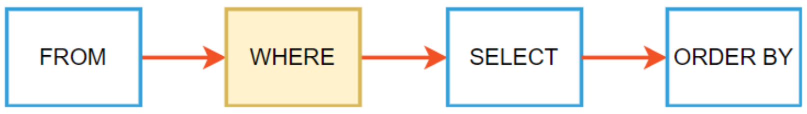
The `search_condition` is a combination of one or more expressions using the logical operator `AND` , `OR` and `NOT` .

In MySQL, a predicate is a Boolean expression that evaluates to `TRUE` , `FALSE` , or `UNKNOWN` .

The `SELECT` statement will include any row that satisfies the `search_condition` in the result set.

Besides the `SELECT` statement, you can use the `WHERE` clause in the `UPDATE` or `DELETE` statement to specify which rows to update or delete.

When executing a `SELECT` statement with a `WHERE` clause, MySQL evaluates the `WHERE` clause after the `FROM` clause and befo the `SELECT` and `ORDER BY` clauses:



AND Operator:

MySQL doesn't have a built-in Boolean type. Instead, it uses the number zero as FALSE and non-zero values as TRUE.

The `AND` operator is a logical operator that combines two or more Boolean expressions and returns 1, 0, or NULL:

```
A AND B
```

In this expression, A and B are called operands. They can be literal values or expressions.

The logical AND operator returns 1 if both A and B are non-zero and not NULL. It returns 0 if either operand is zero; otherwise, it returns NULL.

The logical AND operator returns 1 if both A and B are non-zero and NOT NULL.

OR Operator:

The MySQL `OR` operator is a logical operator that combines two `Boolean` expressions.

```
A OR B
```

If both A and B are not NULL, the OR operator returns 1 (true) if either A or B is non-zero.

IN Operator:

The IN operator allows you to determine if a value matches any value in a list of values. Here's the syntax of the IN operator:

```
value IN (value1, value2, value3,...)\
```

The IN operator returns 1 (true) if the value equals any value in the list (value1 , value2 , value3 ,...). Otherwise, it returns 0.

In this syntax:

- First, specify the value to test on the left side of the IN operator. The value can be a column or an expression.
- Second, specify a comma-separated list of values to match in the parentheses.

BETWEEN Operator:

The BETWEEN operator is a logical operator that specifies whether a value is in a range or not. Here's the syntax of the BETWEEN operator:

```
value BETWEEN low AND high;
```

The BETWEEN operator returns 1 if:

```
value >= low AND value <= high
```

Otherwise, it returns 0.

If the value , low , or high is NULL , the BETWEEN operator returns NULL .

For example, the following statement returns 1 because 15 is between 10 and 20:

```
SELECT 15 BETWEEN 10 AND 20;
```

The following example returns 0 because 15 is not between 20 and 30:

```
SELECT 15 BETWEEN 20 AND 30;
```

IS NULL Operator:

To test whether a value is NULL or not, you use the IS NULL operator. Here's the basic syntax of the IS NULL operator:

```
value IS NULL
```

If the value is NULL , the expression returns true. Otherwise, it returns false.

Note that MySQL does not have a built-in BOOLEAN type. It uses the TINYINT(1) to represent the BOOLEAN values i.e., true means 1 and fals
means 0.

Because the IS NULL is a comparison operator, you can use it anywhere that an operator can be used e.g., in the SELECT or WHERE clause.

See the following example:

```
SELECT 1 IS NULL,  -- 0
       0 IS NULL,  -- 0
       NULL IS NULL; -- 1
```

To check if a value is not NULL , you use IS NOT NULL operator:

```
value IS NOT NULL
```

This expression returns true (1) if the value is not NULL . Otherwise, it returns false (0).

Conclusion:

In this session, we have learnt about:

- how to open the command line for mySQL server using folder approach or through XAMPP.
- creating or droping of databases and tables.

- main statements, conditional and operators in mySQL.

Interview Questions:

How do you create a database in MySQL?

> Use the following command to create a new database called ‘books’:

```
`CREATE DATABASE books;`
```

How do you create a table using MySQL?

~~~
>Use the following to create a table using MySQL:

```sql
CREATE TABLE history (
author VARCHAR(128),
title VARCHAR(128),
type VARCHAR(16),
year CHAR(4)) ENGINE InnoDB;
~~~
```

###How do you Insert Data Into MySQL?

```
>The INSERT INTO statement is used to add new records to a MySQL table:

```sql
INSERT INTO table_name (column1, column2, column3,...)
VALUES (value1, value2, value3,...)
```

If we want to add values for all the columns of the table, we do not need to specify the column names in the SQL query.
```

```
INSERT INTO table_name
VALUES (value1, value2, value3, ...);
```

Thank You !
```