# DATABASE CONCEPTS AND MYSQL

## INTRODUCTION

### WHAT IS A DATABASE?

A database is an organized collection of structured information, or data, typically stored electronically in a computer system.

### WHAT IS A DBMS?

A DBMS serves as an interface between the database and its end users or programs, allowing users to retrieve, update, and manage how the information is organized and optimized.

In other words, we can describe a DBMS (Database management System) as a software that can be used to create and manage databases. The DBMS serves as an interface between the database and end users or application programs.

### WHAT IS SQL?

SQL is a programming language used by nearly all relational databases to query, manipulate, and define data, and to provide access control.

### WHAT IS MYSQL?

MySQL is an open-source relational database management system based on SQL. It was designed and optimized for web applications and can run on any platform.

## DATABASE CONCEPTS

### DATABASE SCHEMA

Database Schema is the design of a database. It can be called as the blueprint to the structure of the database. It tells us how data is organised and stored into the database.

#### TYPES OF DATABASE SCHEMAS

There are three mainly,

- Relational Database Schema
- Network Database Schema
- Hierarchical database schema

#### RELATIONAL DATABASE SCHEMA

It is a model in which data points kept related to each other. The data is represented in tables. In this schema, the columns hold the attributes of the data and the rows holds the data.

#### NETWORK DATABASE SCHEMA

It is a database design in which a record/object has multiple parent data and can have multiple Childs too. The pointers which establish relation between the objects is termed as a link. These links can establish relationship between multiple objects.

#### HIERARCHICAL DATABASE SCHEMA

Here, the records are organised as trees . Here, a single link is made between a parent record and its child record.

### DATA CONSTRAINT

The restriction put on an attribute of the data is termed as a data constraint.

### META-DATA

A data which describes about another data is termed as meta data. The database schema along with various constraints on the data is stored by DBMS in a database catalogue or dictionary, called meta-data.

### DATABASE INSTANCE

An instance is the combination of software and memory that act upon (access or modify) the data in the database.

### QUERY

A query is a request to a database for obtaining information in a desired way.

## RELATIONAL DATABASE CONCEPTS

### ATTRIBUTE

Characteristic or parameters for which data are to be stored in a relation.

In other words, they are the properties that define all the items in the row.

### TUPLE/ RECORD

It is the row in which data is stored.

A tuple is a relationship between the “N” related values.

## DOMAIN

It is a set of values from which an attribute can take a value.

It defines the data type that an attribute can hold.

## DEGREE

The number of attributes (columns) that a relation holds is termed as degree of the relation.

## CARDINALITY

The number of entries / records/ tuples that a relation holds is termed as the cardinality of the relation.

## KEYS

There is a property in SQL that no tuples in a relation must be identical. So, there must be distinct tuples. So, there must be an attribute which is distinct and not null. This attribute helps to distinctly identify a tuple and is known as a key.

A key can be defined as " An attribute which uniquely identify the tuples."

There are following types of keys,

1) Primary key 2) Candidate Key

3) Alternate key 4) Foreign Key

5) Composite Primary Key

### PRIMARY KEY

It is a set of one / more attributes which is used to uniquely identify values from a table.

### CANDIDATE KEY

It is a set of one/ more attributes which are able to uniquely identify values from the table. The one of these which is used to uniquely identify unique values from the table is termed as primary key.

### ALTERNATE KEY

A candidate key which is not the primary key is termed as alternate key.

### FOREIGN KEY

A non- key attribute of a table which is a primary key for another table is termed as a foreign key.

foreign key is an attribute whose value is derived from the primary key of another relation.

### COMPOSITE PRIMARY KEY

If no single attribute in a relation is able to uniquely distinguish the tuples, then more than one attribute is taken together as primary key.

Such primary key consisting of more than one attribute is called Composite Primary key.

# SQL DATATYPES AND THEIR CONSTRAINTS (MYSQL)

## DATA TYPES IN MYSQL

| **Data Type** | **Description** |
|---|---|
| *NUMERICAL DATA TYPES* | |
| INT/ INTEGER | Integer data type (But never use it for accepting phone numbers since it cannot handle 10 digit large numbers)<br>(Can accept Negative values) |
| BIGINT | For big integers (better for phone numbers) |
| DECIMAL (X,Y) | It is used to store decimal Values.<br>X = No of total digits<br>Y= No of decimal points<br>Example:<br>DECIMAL (5,2)-> xxx . xx<br>So, X-Y integer value places… |
| *Date and Time Datatypes* | |
| DATE | Stores Data in format YYYY-MM-DD |
| TIME | Usually Stores time in format hh :mm : ss |
| TIMESTAMP | Accepts value as timestamp including the Time zone<br>yyyy -mm-dd hh :mm :ss |
| String Data Types | |
| CHAR(length) | Declares space for fixed characters. The spaces are kept intact even if the whole space is not used.<br>If the given input during the insertion of values exceeds the amount acceptable, it causes an error. |
| VARCHAR(L) | It declares space upto the specified characters. But it frees up the unused spaces. |
| Other Data Types | |
| BLOB | Standing for Binary Large Objects, it can store large data up to 65535 characters |

| | |
|---|---|
| MEMO/LONG | Can store up to 2 Gb characters or remarks |

### CONSTRAINTS IN SQL

There are constraints that can be passed while declaring the attributes. The attributes may have following constraints.

| Constraint | Description |
|---|---|
| NOT NULL | Makes sure that the attribute never gets null value when adding values to the record. |
| UNIQUE | Ensures that every item in the row has unique items |
| DEFAULT | Makes a default value for the attribute<br>Syntax:<br>default <what is the default value?> |
| PRIMARY KEY | The column which can uniquely identify each row/record in a table. It is a combination of NOT NULL and UNIQUE |
| FOREIGN KEY | The column which refers to value of an attribute defined as primary key in another table |

## MYSQL QUERIES CLASSIFICATION

The Queries can be classified into four heads namely,

1) DDL- Data Definition Language
2) DQL- Data Query Language
3) DCL- Data Control Language
4) DML- Data Manipulation Language

### DATA DEFINITION LANGUAGE

The commands which define or modify the schema of the database are termed as DDLs.

Examples:

CREATE, ALTER, DROP, RENAME. TRUNCATE and COMMENT

### DATA QUERY LANGUAGE

The commands which query the Database. It is nothing else than the SELECT statement.

### DATA MANIPULATION LANGUAGE

The commands which manipulate (manage or influence ) the data are termed as DMLs.

Examples: INSERT, DELETE, UPDATE, EXPLAIN, LOCK TABLES etc.

### DATA CONTROL LANGUAGE

They are the commands which are concerned with authorisation and rights of users of the DB. Examples are GRANT, REVOKE.

### TCL- TRANSACTION CONTROL LANGUAGE

Deals with transactions to the DB. Examples COMMIT, ROLLBACK and SAVEPOINT.

## MYSQL QUERIES FOR DATA DEFINITION

In order to work with data, we first need to create a schema. So, let's check the statements for creating the schemas first.

### CREATE DATABASE <DB_NAME>; AND CREATE TABLE <TABLE_NAME>();

### CREATING DATABASE

**CREATE DATABASE <DB>** is the command for creation of the database.

Example:

CREATE DATABASE my_data_base;

However, in other web development, it is sophisticated, and a single click can create a database and you can directly get into it.

### CREATING TABLES

**CREATE TABLE <TN>();** is the command for creating tables. During declaration, we need to give datatypes and optionally the constraints.

Example

CREATE TABLE

Syntax:

CREATE TABLE sample(s_no **int primary key**, name **varchar(255)**);

### CHECKING THE LIST OF DATABASES AND TABLES

SHOW DATABASES;

The above command is used to show the list of databases.

SHOW TABLES

The above command shows the tables in the database.

### USING THE DATABASE

**USE DATABASE;** command is used to use the database for manipulating or creation of data.

Example: USE my_database;

### DELETE ALL ENTRIES IN A TABLE WITHOUT CHANGING THE STRUCTURE OF THE TABLE

Below command is used to delete the items in the table without modifying the structure of the table.

**TRUNCATE <TABLENAME>**;

### DELETING THE DATABASE

To permanently delete the database, we use the statement,

**DROP DATABASE <DB_NAME>**

Example:

DROP DATABASE MYDB;

### DELETING THE TABLE

To delete a table, we use the statement,

DROP TABLE <TABLE_NAME>;

Example:

DROP TABLE my_table;

### RENAMING THE TABLE

We use the following statement to rename the table

**RENAME TABLE <OLD_TN> TO <NEW_TN>**

Example:

RENAME TABLE student TO my_student;

### ALTERING THE STRUCTURE OF THE TABLE

To alter the structure and schema of the table, we use the **ALTER** statement.

#### ADDING COLUMNS/ATTRIBUTES TO THE TABLE

We use the statement,

**ALTER TABLE <TN> ADD COLUMN <CN> < datatype and constraints > ;**

#### DELETING AN ATTRIBUTE

To delete an attribute, we use the statement,

**ALTER TABLE <tn>DROP COLUMN <cn> ;**

#### CHANGING THE DATATYPE OF A COLUMN

FYI: You cannot change a string to a numerical one. But you can change a numerical one to a string data type.

We use the statement

**ALTER TABLE <tn> MODIFY COLUMN <cn> <type and constraint>;**

#### RENAMING A COLUMN

To rename a column, we use

**ALTER TABLE <t_n> RENAME <cols> TO <new_cols> ;**

## INSERTING VALUES INTO THE TABLES

#### CASE-1 – WHEN ALL VALUES CAN BE MODIFIED AND ALL VALUES ARE GETTING ENTERED

Then we use the following syntax:

**INSERT INTO <T_N> VALUES (a,b,c),(d,e,f).....;**

But, it is to be ensured that the corresponding data type is correct. And we must give the exact column number.

i.e., Len(columns) == Len(values inputted)

**Example:**

Let a table “abcde” have 3 columns with 2 int and 1 str.

INSERT INTO abcde VALUES (1,2,”Name”);

If you just give 1 and 2 it would cause an error

#### CASE-2 YOU NEED TO ENTER VALUES ONLY TO SPECIFIED COLUMNS

We can use the following syntax.

**INSERT INTO <T_N> (col1, col2...) VALUES (val1, val2..) ;**

**Example :**

INSERT INTO abcde(col1,col2) VALUES (1,2) ;

## UPDATING/CHANGING ENTRIES IN A TABLE

It is a combination of multiple words. The syntax is,

**UPDATE <T_N> SET col1=val ,col2=val.. WHERE <condition> ;**

## DELETING PARTICULAR ENTRIES

We use the following syntax,

**DELETE FROM <TN> WHERE <Conditional>;**

## FINDING THE STRUCTURE OF THE TABLE

To find the structure of the table, we use

**DESCRIBE/DESC <T_N>**

## QUERYING THE DATABASE

To Query the database, we use the following syntax,

**SELECT <col-names> FROM <TN> WHERE <Condition>;**

To select everything, we use the asterisk (*).

### USING DIFFERENT NAMES DURING QUERY

To use difference name of table or column during query we use the operator **AS.**

For Column

**SELECT * AS <name> FROM <tn> WHERE <cond>;**

For Table:

**SELECT <cols> FROM <t_n> AS <name> WHERE <cond>;**

### SELECTING DISTINCT ITEMS

To select distinct items, we use DISTINCT clause.

Syntax:

**SELECT distinct <item> FROM <TN> WHERE<cond>;**

### ORDERING THE RESULTS ACCORDING TO A CONDITION

To order the result according to some conditions, we use the ORDER BY Clause.

Syntax:

**SELECT <items> FROM <TN> <if needed use WHERE CLAUSE> ORDER BY item1 <ASC|DESC>, item2 <ASC/DESC>**

By default, ORDER BY orders in ascending manner. If needed, you can explicitly specify DESC.

You can even order by two or more conditions. If an attribute has same value, the next attribute's ORDER by is used up.

### GROUPING THE REQUISITES

To group the items which have a common attribute, we use the **GROUP BY** Clause.

NOTE: You cannot use WHERE after GROUP BY. You need to use HAVING clause instead.

Syntax for group by:

**SELECT <items including at least one aggregate fn> FROM <tn> WHERE(optional) GROUP BY <attribute>**

Syntax for selective group by:

**SELECT <items including at least one aggregate fn> FROM <tn> WHERE(optional) GROUP BY <attribute> HAVING Aggfn(attribute)**

### USING THE CONDITIONALS

There are many ways to use the conditionals for different data types.

#### USING COMAPARISIONS IN CONDITIONS

After the where clause, we can use
a =b # In python it is == in SQL it is just =

| Operator | Description |
|---|---|
| A=B | Checks if the values are equal (in python it is == whereas in SQL it is =) |
| A>B | A greater than B |
| A<B | A less than B |
| A>=B and A<=B | A greater than or equal to B and A less than or equals B |

#### CONDITIONALS BASED ON RANGE

We use the statement,

**BETWEEN <starting value> AND <ending value>**

If we need do not need the value between them, we use NOT BETWEEN.

#### CONDITIONALS BASED ON A SEQUENCE

We use the statement,

**<valuetocheck> IN <array of sequence>;**

### USING PATTERNS

We use the like operator to check the patterns.
Syntax:

**<attribute>LIKE <pattern>**

## GETTING PATTERNS IN A STRING

### EXACT SPACES

To get patterns which have 5th character as "D", we write

<attribute> LIKE "____D"

Here, 4 underscores are given to signify that 5th position is "D".

### ANY LENGTH BEFORE AND AFTER

To get patterns which have D as last letter, we write,

<attribute> LIKE "%D".

% Is symbol for any acceptance of characters.

If we want to get patterns where a letter "a" we use "%a%".

## AGGREGATE FUNCTIONS IN SQL

They are the ones which act upon the columns of the data set returned.

| Function syntax | Result |
|---|---|
| AVG(COL) | Returns the average |
| SUM(col) | Returns the sum |
| MAX(col) , MIN(col) | Returns the max and min value in the column |
| COUNT(col) | Returns the rows in the column |

## MYSQL AND PYTHON CONNECTIVITY

### ALGORITHM:

- Make a connection to the database.
- Create cursor.
- Do the changes and commit them.
- Close the connection.

### MODULE TO BE USED

Distribution name- mysql-connector-python-8.0.23

Module name-> mysql.connector

### GETTING STARTED INTO CONNECTOR

- Import mysql.connector
- Initialise connection object as follows

**Con_obj= mysql.connector(host="hostname", user="username",passwd="password")**
Usually, host is local host, if remote connection is used, you can change them. User is usually root, but you can have other names too.

- Initialise the cursor variable that would give access to the database.
**Cursor= con_obj.cursor()**
- To execute a command, we use **cursor.execute("statement in string")**
- To fetch the returned items from select statement we use **cursor.fetchall()** statement.
- To get the result, we use **cursor.fetchone()** statement.
- TO NOTE: WE DO NOT GET THE COLUMN NAMES. To get them, we must use **cursor.column_names**
- The column name is just a tuple. It is not a function. So, parenthesis must not be given.
- To get no of rows in the cursor space, we use **cursor.rowcount**
- **Con_obj.rollback()** method is used to rollback the changes done by the current connection (current txn)
- Usually, if any problem rises, the txn is rollbacked. If you don't want it, you can manually commit the things using con_obj.commit() method
- To close the connection, we use con_obj.close and to close the cursor, we use cursor.close()