→ MySQL

a. MySQL is Structured Query Language and SQL commands are further divided into 4 sub categories

i. DDL (Data Definition Language)

1. DDL is used to modify any database or table structure and schema
2. These statements are used to handle storage of database objects for designing
3. CREATE, ALTER, DROP, TRUNCATE commands fall under this category of commands

ii. DQL (Data Query language)

1. DQL statements are used for performing queries on the data within schema objects.
2. The purpose of the DQL Command is to get some schema relation based on the query passed to it.
3. It is an SQL statement that allows getting data from the database and imposing order upon it.
4. It includes the SELECT statement.
5. This command allows getting the data out of the database to perform operations with it.
6. When a SELECT is fired against a table or tables the result is compiled into a further temporary table, which is displayed or perhaps received by the program i.e. a front-end.

iii. DML (Data Manipulation Language)

1. DML statements affect the records in the table and perform basic operations like selecting any record, inserting any new record, deleting any record or updating or modifying existing records
2. SELECT, INSERT, UPDATE, DELETE Commands fall under this category of commands

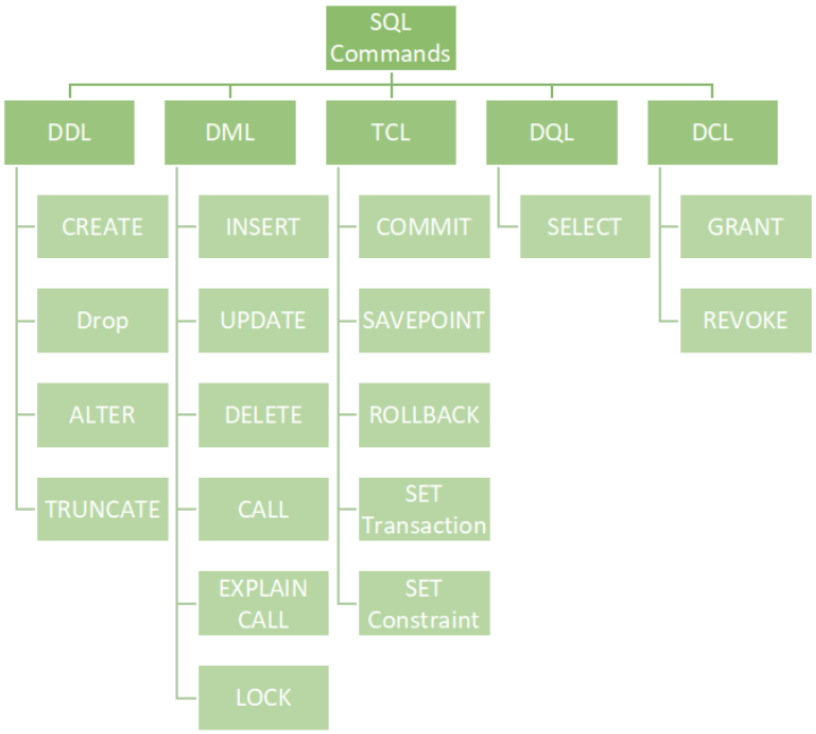
iv. DCL (Data Control Language)

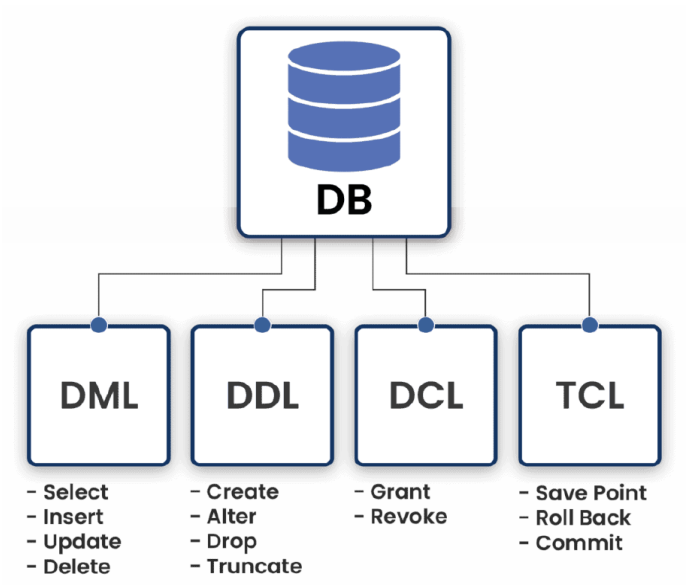
1. DCL commands control the level of access that users have on database objects
2. Commands like GRANT(allows the user to read/write on a certain database). REVOKE(taking away the already granted permission to read/write on database object) fall under this category of commands

v. TCL (Transaction Control Language)

1. TCL statements allows you to control and manage transaction to maintain the integrity of data within the system

2. Commands like BEGIN(this opens the transaction), COMMIT(to commit any transaction), ROLLBACK(to undo/rollback any transaction in case of any error) fall under this category of commands

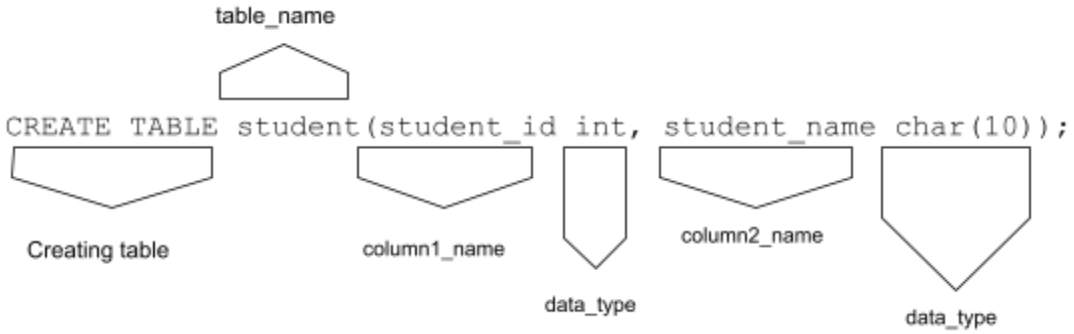




→ Data Types in MySQL

a. A data type is a type/category to which data belongs

b. Data type defines the type of data to be stored in each column of the table
c. Each column/attribute has its own type, we need to specify the type during creation of any table
d. Syntax:
```
CREATE TABLE table_name(column1 <dataType>, column2 <dataType>);
```
e. Example:
```
CREATE TABLE student(student_id INT, Student_name CHAR(30));
```



f. String and Character data types:

i. CHAR

1. This data type can hold alphabets, numbers & special symbols
2. This data type is of fixed length as specified at the time of creation of variable
3. If length of data type is not specified, then default size is 1 character
4. It'll occupy memory according to the size of parameter
5. Suppose , `CHAR(10)` is specified as data type, it'll occupy 10 bytes of memory
6. The length of char can vary from 0 to 255
7. Length is fixed when we declare a table

ii. VARCHAR

1. It stands for `VAR`iable `CHAR`acter, which means, it can store variable length of string
2. The range of characters this can hold is between 0 to 65,535
3. You cannot exceed the length of string defined using size parameter
4. Its default size is also 1
5. Suppose you've declared `VARCHAR(20)`, and you've assigned a value `"Ram"`, as size of `"Ram"` is 3 bytes, it'll occupy only 3 bytes and release other spaces, thus it'll not pad unoccupied bytes

Interview Question: CHAR Vs. VARCHAR

VARCHAR	CHAR
It is variable length data type	It is fixed length data type
No padding (white-space) is necessary, as it is of variable size	Padding is automatically put at the right side to store the string when its length is less than declared length
Length can range between 0 to 65,535	Length can range between 0 to 255

iii. BLOB

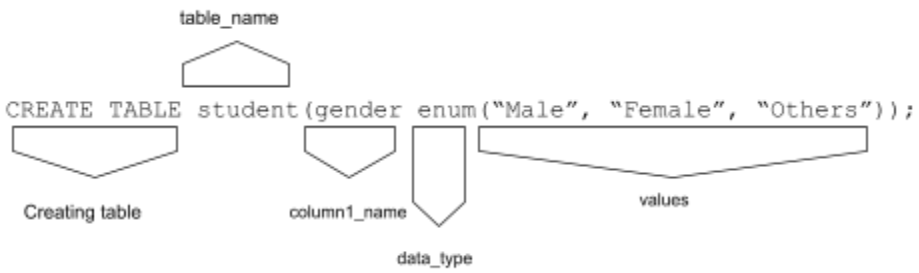
1. It stands for Binary Large OBject
2. This can hold variable amount of data
3. It can store binary data such as image, pdf, videos, etc
4. Suppose, your data contains 50 bytes, then BLOB will occupy 52 bytes, as it'll add 2 bytes of data overhead to each specified data

iv. TEXT

1. TEXT is useful for storing long format text String such as articles, blogs, etc
2. It can hold data up-to 4GB in length
3. Suppose, your data contains 50 bytes, then TEXT will occupy 52 bytes, as it'll add 2 bytes of data overhead to each specified data

v. ENUM

1. This is a string object whose value is chosen from a list of permitted values defined at the time of table creation
2. This provides compact storage of data
3. If a value is assigned which is not in the list, a blank value will be assigned
4. Syntax:
   ```
   CREATE TABLE table_name(column_name enum("val1", "val2", "val3"));
   ```
5. Example:
   ```
   CREATE TABLE student(gender enum("Male", "Female", "Others"));
   ```



g. Numeric data types

i. **INT(size)**
   1. It is used for storing integer type values
   2. The size parameter specifies the maximum length of the number which you can store
   3. Default size of `INT` is 4 bytes, `BIGINT` is 8 bytes

ii. **FLOAT(p)**
   1. Here 'p' is used to specify whether to use float or double for the resulting data type
   2. If the value of 'p' ranges from 0-24, then it is float
   3. If the value of 'p' ranges from 25-53, then it is double
   4. The default size of `FLOAT` is 4 bytes

iii. **FLOAT(size, d)**
   1. The length of digits is specified using 'size' parameter
   2. The number of digits after decimal point is specified by 'd' parameter
   3. The default size of `FLOAT` is 4 bytes
   4. Example: `FLOAT(3, 2)` means that you can store the size of the number till 3 and 2 digits after decimal

iv. **DOUBLE(size, d)**
   1. The length of digits is specified using 'size' parameter
   2. The number of digits after decimal point is specified by 'd' parameter
   3. The default size of `DOUBLE` is 8 bytes
   4. Example: `DOUBLE(3, 2)` means that you can store the size of the number till 3 and 2 digits after decimal

v. **BOOLEAN**
   1. False is stored as '0' and True is stored as '1'

vi. **DATE**
   1. You can store data in `"YYYY-MM-DD"` format
   2. Date within the range `"1000-01-01"` to `"9999-12-31"` can be stored

vii. **TIME**
   1. It is used for storing the time of a day
   2. It can store time in format `"HH-MM-SS"`

viii. **DATETIME**
   1. It can store date & time altogether in format `"YYYY-MM-DD HH-MM-SS"`

ix. **TIMESTAMP**
   1. It stores the data in the same format as `DATETIME`

2. You can automatically update/store the current date & time by using `default current_timestamp`

x. YEAR

1. It is used to store year in 4-digit format "`YYYY`"

INT	4 bytes
BIGINT	8 bytes
FLOAT	4 bytes
DOUBLE	8 bytes

→ Starting to operate on Database

a. `SELECT`:
   i. We can view a table using `SELECT` statement
   ii. Here `'*'` will give the output, all of the records along with their attributes will be displayed
   iii. Syntax:
   ```
   SELECT * FROM table_name;
   ```
   iv. Example:
   ```
   SELECT * FROM employees;
   ```

b. `CREATE DATABASE`:
   i. To use any data from a table, first we need to create a database using `SELECT` command
   ii. Syntax:
   ```
   CREATE DATABASE database_name;
   ```
   iii. Example:
   ```
   CREATE DATABASE hr;
   ```

c. `USE`:
   i. If database is already created, then we need to use the database command as
   Syntax:
   ```
   USE database_name;
   ```
   Example:
   ```
   USE hr;
   ```

d. `DESC`:
   i. We can see the description of any table using DESC command

ii. Syntax:
```
DESC table_name;
```
iii. Example:
```
DESC employees;
```

→ SELECT statement

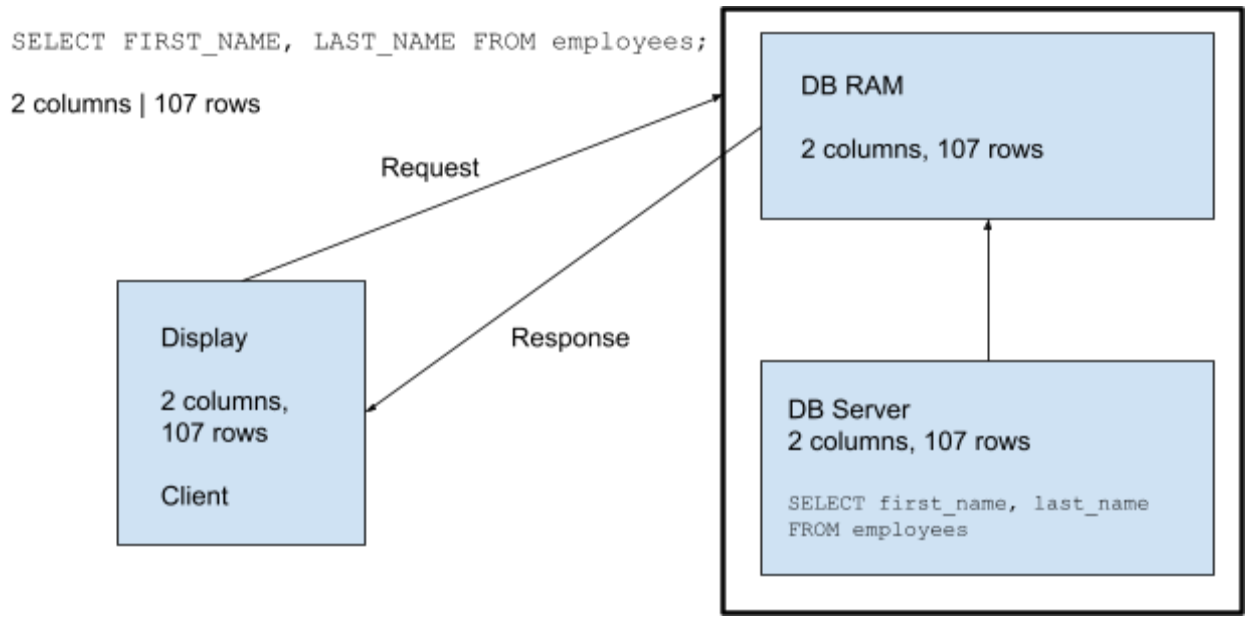
a. `SELECT * FROM table_name;`
b. To use any data from a table, first we need to create a database using `SELECT` command

→ DESC statement

a. `DESC table_name;`
b. It is used to see the description of any table
c. It gives you the structure or schema of any table which includes column/attribute names, data types, constraints, indexes applied on it

→ SHOW statement

a. Alternative way to show the structure of table/database, we can use
```
SHOW COLUMNS FROM table_name;
```
b. To display all the tables in current database
c. To use this command, firstly we need to use the database, and then show tables from database
```
USE hr;
SHOW TABLES;
```
d. Alternate way to show table is
```
SHOW TABLE STATUS;
```
e. To show databases, we use
```
SHOW DATABASES;
SHOW DATABASES LIKE %hr%; #works in different version
```



→ Alias

a. Aliases are used to give temporary names to any column or table
b. It is used to make a statement more readable
c. We can create an alias using `'AS'` keyword, for any table or any column

d. Syntax (column Alias - using `AS` keyword):
```
SELECT colun_name AS alias_name FROM table_name;
```
Example (column Alias - using `AS` keyword):
```
SELECT FIRST_NAME AS fname FROM employees;
```

e. Syntax (Table Alias):
```
SELECT <table_alias_symbol>.<column_name> FROM <table_name>
<table_alias_symbol>;
```
Example (Table Alias):
```
SELECT e.FIRST_NAME FROM employees e;
```

f. Syntax (Shorthand Alias - without using `AS` keyword):
```
SELECT column_name alias_name FROM table_name;
```
Example (Shorthand Alias - without using `AS` keyword):
```
SELECT FIRST_NAME fname FROM employees;
```

g. Example (Temporary View Using Alias):
```
SELECT FIRST_NAME AS fname, salary*10 increment, salary FROM
employees;
```