## ➔ What is Database?

A database is an application that stores the organized collection of records. It can be accessed and manage by the user very easily. It allows us to organize data into tables, rows, columns, and indexes to find the relevant information very quickly. Each database contains distinct API for performing database operations such as creating, managing, accessing, and searching the data it stores. Today, many databases available like MySQL, Sybase, Oracle, MongoDB, PostgreSQL, SQL Server, etc. In this section, we are going to focus on MySQL mainly.

## ➔ Differences

### DBMS vs RDBMS vs NoSQL

| Parameter | DBMS | RDBMS | NoSQL |
|---|---|---|---|
| Storage | Data stored as a file system. | RDBMS applications store data in the form of table structured manner. | NoSQL is a non-relational database system. It stores data in the form of unstructured. manner. |
| Number of users | It supports a single user. | RDBMS supports multiple users. | It also supports multiple users. |
| Database structures | In DBMS, data stores either in either a navigational or hierarchical form. | RDBMS uses tabular structures to store data. In table headers are the column names and the rows contains corresponding values. | NoSQL uses to store data in structured, semi-structured and unstructured forms. |
| ACID | In regular DBMS, the data not be stored following the ACID. It creates an inconsistencies database. | RDBMS are harder to construct and obey ACID (Atomicity, Consistency, Isolation, Durability). It helps to create consistency database. | NoSQL may support ACID to store data. |

| | | | |
|---|---|---|---|
| Normalization | The database does not support normalization. | It supports the normalization and joining of tables. | It does have table form, so it does not support normalization. |
| open-source | It's a mix of an open-source and commercial database. | Open-source application | Open-source program |
| Integrity constraints | DBMS database does not support the integrity constants. | The relational database supports the integrity constraints at the schema level. Data values beyond a defined range cannot be stored in the particular RDBMS column. | NoSQL database supports integrity constraints. |
| Development year | In the 1960s DBMS is introduced to store data. | It was developed in the 1970s to deal with the issues of flat file storage. | It developed in the late 2000s to overcome the issues and limitations of the SQL database. |
| Distributed database | It does not support a distributed database. | It supports a distributed database. | It also supports a distributed database. |
| Ideally suited for | This system deals with a small quantity of data. | This database system deals with a large quantity of data. | NoSQL database mainly designed for Big data and real-time web data. |
| Client-server | DBMS system does not support client-server architecture. | RDBMS program support client-server architecture. | NoSQL storage system supports multi-server. It also supports client-server architecture. |
| Data relationship | No relationship between the data value | Data related to each other with the help of foreign keys | Data can be stored in a single document file. |
| Hardware and software | Low software and hardware | High software and specialized DB hardware (Oracle Exadata, etc.) | Commodity hardware |
| Data fetching | Data fetching is slower for complex data. | Data fetching is rapid because of its relational approach and database. | Data fetching is easy and flexible. |
| Examples | XML, Windows Registry, file system, etc. | MYSQL, Oracle, SQL Server, etc. | Apache HBase, IBM Domino,Oracle NoSQL Database,etc. |

## ➔ MySQL Versions

MySQL 5.5, 5.6, 5.7, 8.0

# ❖ SQL:

- ➔ SQL is the standard language for dealing with Relational Databases.
- ➔ SQL is used to insert, search, update, and delete database records.
- ➔ Example:

  SELECT * FROM Customers;

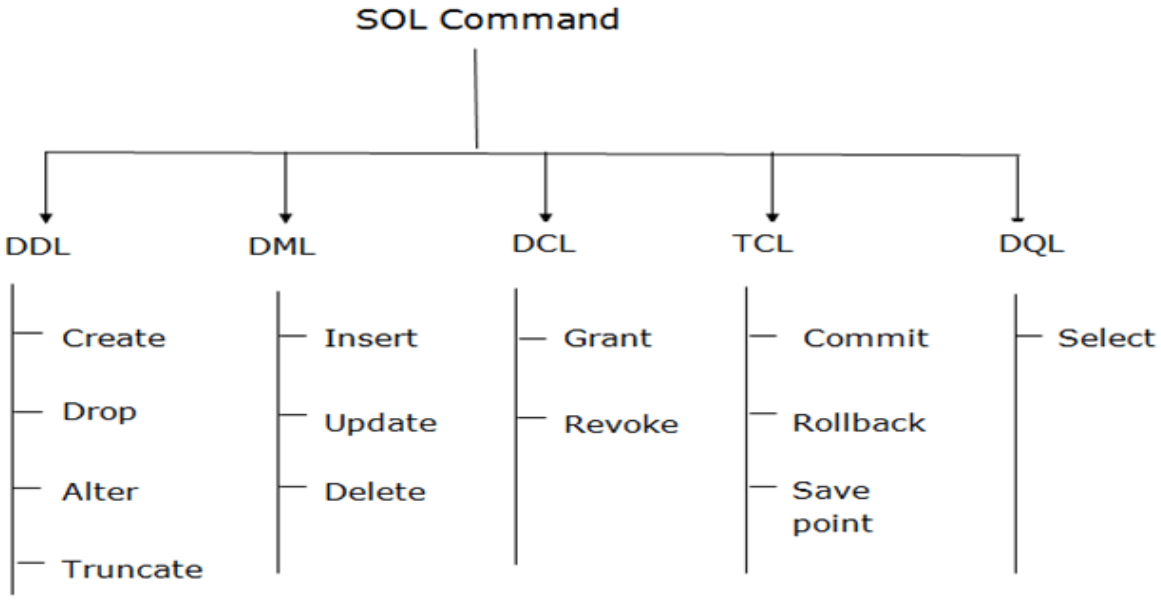
- ➔ Not case sensitive
- ➔ Semicolon at the end

## ❖ 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.




## 1. 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

## 2. 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.

- INSERT

- UPDATE
- DELETE

## 3. Data Control Language

DCL commands are used to grant and take back authority from any database user.

- Grant
- Revoke

## 4. Transaction Control Language

TCL commands can only use with DML commands like INSERT, DELETE and UPDATE only.

These operations are automatically committed in the database that's why they cannot be used while creating tables or dropping them.

- COMMIT
- ROLLBACK
- SAVEPOINT

## 5. Data Query Language

DQL is used to fetch the data from the database.

- SELECT

→ SQL Constraints

Constraints are the rules enforced on data columns on a table. These are used to limit the type of data that can go into a table. This ensures the accuracy and reliability of the data in the database.

Constraints can either be column level or table level. Column level constraints are applied only to one column whereas, table level constraints are applied to the entire table.

Following are some of the most commonly used constraints available in SQL –

- NOT NULL Constraint – Ensures that a column cannot have a NULL value.
- DEFAULT Constraint – Provides a default value for a column when none is specified.
- UNIQUE Constraint – Ensures that all the values in a column are different.
- PRIMARY Key – Uniquely identifies each row/record in a database table.
- FOREIGN Key – Uniquely identifies a row/record in any another database table.
- CHECK Constraint – The CHECK constraint ensures that all values in a column satisfy certain conditions.
- INDEX – Used to create and retrieve data from the database very quickly.

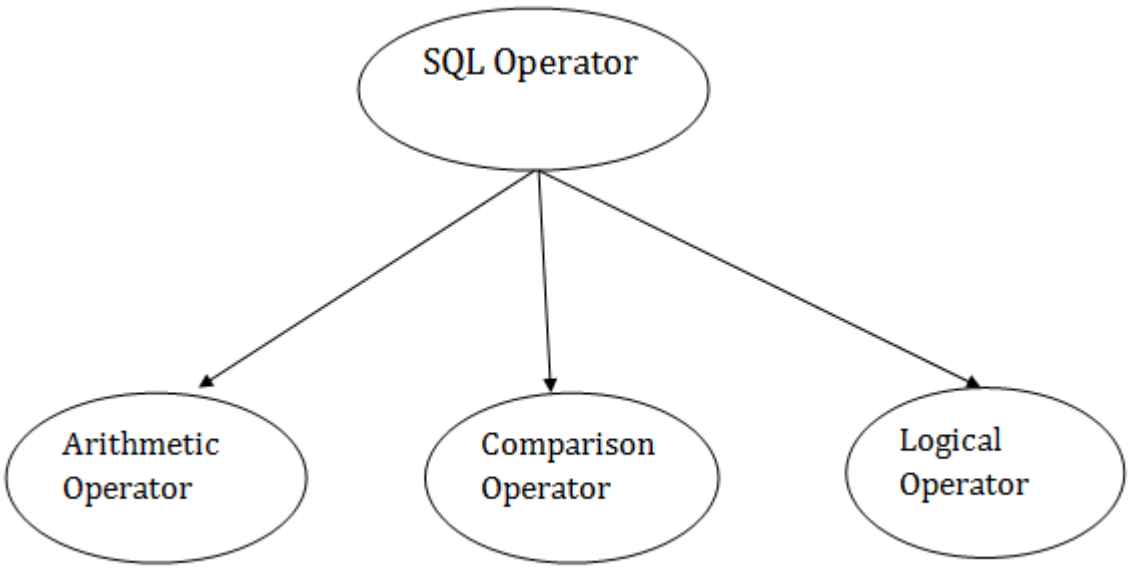
## → Data Integrity

The following categories of data integrity exist with each RDBMS –

- Entity Integrity – There are no duplicate rows in a table.
- Domain Integrity – Enforces valid entries for a given column by restricting the type, the format, or the range of values.
- Referential integrity – Rows cannot be deleted, which are used by other records.
- User-Defined Integrity – Enforces some specific business rules that do not fall into entity, domain or referential integrity.

## ➔ SQL Operators

➔There are various types of SQL operator:




# SQL Arithmetic Operators

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

| Operator | Description | Example |
|---|---|---|
| + | It adds the value of both operands. | a+b will give 30 |
| - | It is used to subtract the right-hand operand from the left-hand operand. | a-b will give 10 |
| * | It is used to multiply the value of both operands. | a*b will give |

| | | |
|---|---|---|
| | | 200 |
| / | It is used to divide the left-hand operand by the right-hand operand. | a/b will give 2 |
| % | It is used to divide the left-hand operand by the right-hand operand and returns reminder. | a%b will give 0 |

## SQL Comparison Operators:

Let's assume 'variable a' and 'variable b'. Here, 'a' contains 20 and 'b' contains 10.

| Operator | Description | Example |
|---|---|---|
| = | It checks if two operands values are equal or not, if the values are queal then condition becomes true. | (a=b) is not true |
| != | It checks if two operands values are equal or not, if values are not equal, then condition becomes true. | (a!=b) is true |
| <> | It checks if two operands values are equal or not, if values are not equal then condition becomes true. | (a<>b) is true |
| > | It checks if the left operand value is greater than right operand value, if yes then condition becomes true. | (a>b) is not true |
| < | It checks if the left operand value is less than right operand value, if yes then condition becomes true. | (a<b) is true |
| >= | It checks if the left operand value is greater than or equal to the right operand value, if yes then condition becomes true. | (a>=b) is not true |
| <= | It checks if the left operand value is less than or equal to the right operand value, if yes then condition becomes true. | (a<=b) is true |

| | | |
|---|---|---|
| !< | It checks if the left operand value is not less than the right operand value, if yes then condition becomes true. | (a!=b) is not true |
| !> | It checks if the left operand value is not greater than the right operand value, if yes then condition becomes true. | (a!>b) is true |

## SQL Logical Operators

There is the list of logical operator used in SQL:

| Operator | Description |
|---|---|
| ALL | It compares a value to all values in another value set. |
| AND | It allows the existence of multiple conditions in an SQL statement. |
| ANY | It compares the values in the list according to the condition. |
| BETWEEN | It is used to search for values that are within a set of values. |
| IN | It compares a value to that specified list value. |
| NOT | It reverses the meaning of any logical operator. |
| OR | It combines multiple conditions in SQL statements. |
| EXISTS | It is used to search for the presence of a row in a specified table. |
| LIKE | It compares a value to similar values using wildcard operator. |

- SELECT - extracts data from a database

```
SELECT * FROM Customers;

SELECT CustomerName, City, Country FROM Customers;

SELECT Country FROM Customers;

SELECT DISTINCT Country FROM Customers;

SELECT COUNT(DISTINCT Country) FROM Customers;
```

➔ SELECT WITH WHERE CONDITION:

```
SELECT * FROM Customers WHERE Country = 'Mexico';

SELECT * FROM Customers WHERE CustomerID = 1;
```

➔ Other Operators in where clause

=, >,<, <=,>=,<>,between, like, in etc.

➔ The MySQL AND, OR and NOT Operators

```
SELECT * FROM Customers
WHERE Country = 'Germany' AND City = 'Berlin';

SELECT * FROM Customers
WHERE City = 'Berlin' OR City = 'Stuttgart';

SELECT * FROM Customers
WHERE Country = 'Germany' OR Country = 'Spain';

SELECT * FROM Customers
WHERE NOT Country = 'Germany';
```

```
SELECT * FROM Customers
WHERE Country = 'Germany' AND (City = 'Berlin' OR City = 'Stuttgart');

SELECT * FROM Customers
WHERE NOT Country = 'Germany' AND NOT Country = 'USA';
```

➔ The MySQL ORDER BY Keyword

```
SELECT * FROM Customers
ORDER BY Country;

SELECT * FROM Customers
ORDER BY Country DESC;

SELECT * FROM Customers
ORDER BY Country, CustomerName;

SELECT * FROM Customers
ORDER BY Country ASC, CustomerName DESC;
```

➢ UPDATE - updates data in a database

```
UPDATE Customers

SET ContactName = 'Alfred Schmidt', City = 'Frankfurt'

WHERE CustomerID = 1;


UPDATE Customers

SET PostalCode = 00000

WHERE Country = 'Mexico';


UPDATE Customers

SET PostalCode = 00000;
```

- DELETE - deletes data from a database

```
DELETE FROM Customers WHERE CustomerName='Alfreds Futterkiste';
```

```
DELETE FROM Customers;
```

### ➔ The MySQL LIMIT Clause

```
SELECT * FROM Customers
LIMIT 3;
```

```
SELECT * FROM Customers
WHERE Country='Germany'
LIMIT 3;
```

### ➔ MySQL MIN() and MAX() Functions

```
SELECT MIN(Price) AS SmallestPrice
FROM Products;
```

```
SELECT MAX(Price) AS LargestPrice
FROM Products;
```

### ➔ MySQL COUNT(), AVG() and SUM() Functions

```
SELECT COUNT(ProductID)
FROM Products;
```

```
SELECT AVG(Price)
FROM Products;
```

```
SELECT SUM(Quantity)
FROM OrderDetails;
```

## ➔ SQL LIKE Examples

```
SELECT * FROM Customers
WHERE CustomerName LIKE 'a%';

SELECT * FROM Customers
WHERE CustomerName LIKE '%a';

SELECT * FROM Customers
WHERE CustomerName LIKE '%or%';

SELECT * FROM Customers
WHERE CustomerName LIKE '_r%';

SELECT * FROM Customers
WHERE CustomerName LIKE 'a__%';

SELECT * FROM Customers
WHERE ContactName LIKE 'a%o';

SELECT * FROM Customers
WHERE CustomerName NOT LIKE 'a%';
```

## ➔ MySQL Wildcards

```
SELECT * FROM Customers
WHERE City LIKE 'ber%';

SELECT * FROM Customers
WHERE City LIKE '%es%';

SELECT * FROM Customers
WHERE City LIKE '_ondon';

SELECT * FROM Customers
WHERE City LIKE 'L_n_on';
```

## ➔ MySQL IN Operator

```
SELECT * FROM Customers
WHERE Country IN ('Germany', 'France', 'UK');

SELECT * FROM Customers
WHERE Country NOT IN ('Germany', 'France', 'UK');

SELECT * FROM Customers
WHERE Country IN (SELECT Country FROM Suppliers);
```

## ➔ The MySQL BETWEEN Operator

```
SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20;

SELECT * FROM Products
WHERE Price NOT BETWEEN 10 AND 20;

SELECT * FROM Products
WHERE Price BETWEEN 10 AND 20
AND CategoryID NOT IN (1,2,3);

SELECT * FROM Products
WHERE ProductName BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;

SELECT * FROM Products
WHERE ProductName BETWEEN "Carnarvon Tigers" AND "Chef Anton's Cajun Seasoning"
ORDER BY ProductName;

SELECT * FROM Products
WHERE ProductName NOT BETWEEN 'Carnarvon Tigers' AND 'Mozzarella di Giovanni'
ORDER BY ProductName;

SELECT * FROM Orders
WHERE OrderDate BETWEEN '1996-07-01' AND '1996-07-31';
```

## ➔ MySQL Aliases

```
SELECT CustomerID AS ID, CustomerName AS Customer
FROM Customers;

SELECT CustomerName AS Customer, ContactName AS "Contact Person"
FROM Customers;

SELECT CustomerName, CONCAT_WS(', ', Address, PostalCode, City,
Country) AS Address
FROM Customers;

SELECT o.OrderID, o.OrderDate, c.CustomerName
FROM Customers AS c, Orders AS o
WHERE c.CustomerName='Around the Horn' AND c.CustomerID=o.CustomerID;

SELECT Orders.OrderID, Orders.OrderDate, Customers.CustomerName
FROM Customers, Orders
WHERE Customers.CustomerName='Around the
Horn' AND Customers.CustomerID=Orders.CustomerID;
```