Sql

- SQL is a standard language for accessing and manipulating databases.
- **What is SQL?**
- SQL stands for Structured Query Language
- SQL lets you access and manipulate databases
- SQL became a standard of the American National Standards Institute (ANSI) in 1986, and of the International Organization for Standardization (ISO) in 1987

What Can SQL do?

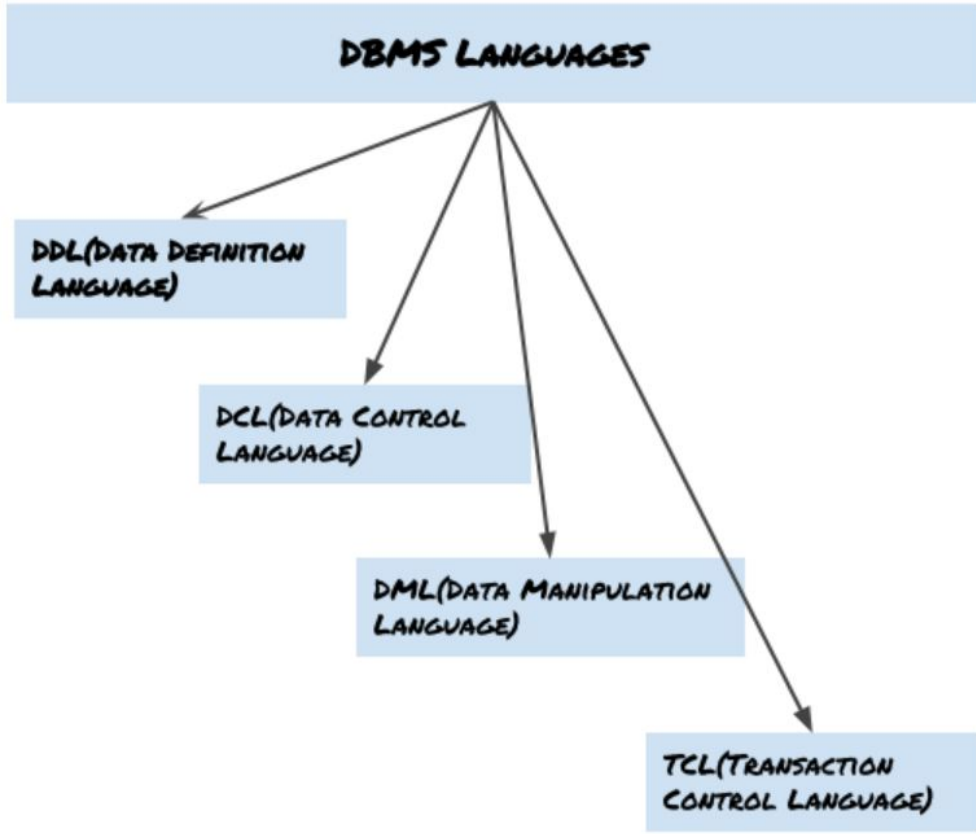
- SQL can execute queries against a database
- SQL can retrieve data from a database
- SQL can insert records in a database
- SQL can update records in a database
- SQL can delete records from a database
- SQL can create new databases
- SQL can create new tables in a database
- SQL can create stored procedures in a database
- SQL can create views in a database
- SQL can set permissions on tables, procedures, and views

RDBMS

- RDBMS stands for Relational Database Management System.
- RDBMS is the basis for SQL, and for all modern database systems such as MS SQL Server, IBM DB2, Oracle, MySQL, and Microsoft Access.
- The data in RDBMS is stored in database objects called tables. A table is a collection of related data entries and it consists of columns and rows.

- Every table is broken up into smaller entities called fields.
- The fields in the Customers table consist of CustomerID, CustomerName, ContactName, Address, City, PostalCode and Country. A field is a column in a table that is designed to maintain specific information about every record in the table.
- A record, also called a row, is each individual entry that exists in a table. A record is a horizontal entity in a table.
- A column is a vertical entity in a table that contains all information associated with a specific field in a table.

Types of Dbms language



DDL

- DDL is used for specifying the database schema. It is used for creating tables, schema, indexes, constraints etc. in database. Lets see the operations that we can perform on database using DDL:
- To create the database instance – **CREATE**
- To alter the structure of database – **ALTER**
- To drop database instances – **DROP**
- To delete tables in a database instance – **TRUNCATE**
- To rename database instances – **RENAME**
- To drop objects from database such as tables – **DROP**
- To Comment – **Comment**
- All of these commands either defines or update the database schema that's why they come under Data Definition language.

Data Manipulation Language (DML)

- DML is used for accessing and manipulating data in a database. The following operations on database comes under DML:
- To read records from table(s) – **SELECT**
- To insert record(s) into the table(s) – **INSERT**
- Update the data in table(s) – **UPDATE**
- Delete all the records from the table – **DELETE**

Data Control language (DCL)

- DCL is used for granting and revoking user access on a database –
- To grant access to user – GRANT
- To revoke access from user – REVOKE

GRANT SELECT, UPDATE ON MY_TABLE TO SOME_USER, ANOTHER_USER;

REVOKE SELECT, UPDATE ON student FROM BCA, MCA;

Transaction Control Language(TCL)

- The changes in the database that we made using DML commands are either performed or rollbacked using TCL.
- To persist the changes made by DML commands in database – COMMIT
- To rollback the changes made to the database – ROLLBACK

Difference between COMMIT and ROLLBACK

	COMMIT	ROLLBACK
1.	COMMIT permanently saves the changes made by the current transaction.	ROLLBACK undo the changes made by the current transaction.
2.	The transaction can not undo changes after COMMIT execution.	Transaction reaches its previous state after ROLLBACK.
3.	When the transaction is successful, COMMIT is applied.	When the transaction is aborted, ROLLBACK occurs.

SQL constraints are used to specify rules for the data in a table.
Constraints 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 table. If there is any violation between the constraint and the data action, the action is aborted.
Constraints can be column level or table level. Column level constraints apply to a column, and table level constraints apply to the whole table.
The following constraints are commonly used in SQL:

- `NOT NULL` - Ensures that a column cannot have a NULL value
- `UNIQUE` - Ensures that all values in a column are different
- `PRIMARY KEY` - A combination of a `NOT NULL` and `UNIQUE`. Uniquely identifies each row in a table
- `FOREIGN KEY` - Prevents actions that would destroy links between tables
- `CHECK` - Ensures that the values in a column satisfies a specific condition
- `DEFAULT` - Sets a default value for a column if no value is specified
- `CREATE INDEX` - Used to create and retrieve data from the database very quickly

SQL NOT NULL Constraint

By default, a column can hold NULL values.
The `NOT NULL` constraint enforces a column to NOT accept NULL values.
This enforces a field to always contain a value, which means that you cannot insert a new record, or update a record without adding a value to this field.

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255) NOT NULL,
    Age int
);
```

SQL UNIQUE Constraint

- The UNIQUE constraint ensures that all values in a column are different.
- Both the UNIQUE and PRIMARY KEY constraints provide a guarantee for uniqueness for a column or set of columns.
- A PRIMARY KEY constraint automatically has a UNIQUE constraint.
- However, you can have many UNIQUE constraints per table, but only one PRIMARY KEY constraint per table.

```
CREATE TABLE Persons (
    ID int NOT NULL UNIQUE,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT UC_Person UNIQUE (ID,LastName)
);
```

SQL PRIMARY KEY Constraint

The PRIMARY KEY constraint uniquely identifies each record in a table. Primary keys must contain UNIQUE values, and cannot contain NULL values.
A table can have only ONE primary key; and in the table, this primary key can consist of single or multiple columns (fields).

```
CREATE TABLE Persons (
    ID int NOT NULL PRIMARY KEY,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int
);
```

Multiple Primary key

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CONSTRAINT PK_Person PRIMARY KEY (ID,LastName)
);
```

SQL FOREIGN KEY Constraint

The `FOREIGN KEY` constraint is used to prevent actions that would destroy links between tables.
A `FOREIGN KEY` is a field (or collection of fields) in one table, that refers to the `PRIMARY KEY` in another table.
The table with the foreign key is called the child table, and the table with the primary key is called the referenced or parent table.
Look at the following two tables:

Persons Table

PersonID	LastName	FirstName	Age
1	Hansen	Ola	30
2	Svendson	Tove	23
3	Pettersen	Kari	20

Orders Table

OrderID	OrderNumber	PersonID
1	77895	3
2	44678	3
3	22456	2
4	24562	1

Notice that the "PersonID" column in the "Orders" table points to the "PersonID" column in the "Persons" table.

SQL CHECK Constraint

The `CHECK` constraint is used to limit the value range that can be placed in a column.
If you define a `CHECK` constraint on a column it will allow only certain values for this column.
If you define a `CHECK` constraint on a table it can limit the values in certain columns based on values in other columns in the row.

SQL CHECK on CREATE TABLE

The following SQL creates a CHECK constraint on the "Age" column when the "Persons" table is created.
The CHECK constraint
ensures that the age of a person must be 18, or older:

MySQL:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    CHECK (Age>=18)
);
```

SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int CHECK (Age>=18)
);
```

SQL DEFAULT on CREATE TABLE

The DEFAULT constraint is used to set a default value for a column.
The default value will be added to all new records, if no other value is specified.

The following SQL sets a DEFAULT value for the "City" column when the "Persons" table is created:

My SQL / SQL Server / Oracle / MS Access:

```
CREATE TABLE Persons (
    ID int NOT NULL,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    City varchar(255) DEFAULT 'Sandnes'
);
```

```
CREATE TABLE Persons (
    Personid int NOT NULL AUTO_INCREMENT,
    LastName varchar(255) NOT NULL,
    FirstName varchar(255),
    Age int,
    PRIMARY KEY (Personid)
);
```

SQL CREATE INDEX Statement

The `CREATE INDEX` statement is used to create indexes in tables.
Indexes are used to retrieve data from the database more quickly than otherwise.
The users cannot see the indexes,
they are just used to speed up searches/queries.

- `CREATE UNIQUE INDEX index_name`
  `ON table_name (column1, column2, ...);`

Sql Query

- Create :

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
```

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    primary key (PersonID)
);
```

Sql Query

- Create :

```
CREATE TABLE table_name (
    column1 datatype,
    column2 datatype,
    column3 datatype,
   ....
);
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    primary key (PersonID)
);
```

```
CREATE TABLE Persons (
    PersonID int,
    LastName varchar(255),
    FirstName varchar(255),
    Address varchar(255),
    City varchar(255),
    Salary numeric(8,2)
    primary key (PersonID)
);
```

Drop table

- `DROP TABLE table_name;`
- `INSERT INTO table_name (column1, column2, column3, ...)VALUES (value1, value2, value3, ...);`
- `SELECT * FROM table_name;`
- `SELECT * FROM Customers WHERE Country='Mexico';`
- `UPDATE CustomersSET ContactName = 'Alfred Schmidt', City= 'Frankfurt' WHERE CustomerID = 1;`
- `DELETE FROM table_name WHERE condition;`

Q. Create a table Student_info as follows and perform the query according to condition as below .

Id	Name	Address	Phoneno	Age	
1	Sharmila Bhattarai	Itahari	984310131	15	
2	Bhumika rai	Biratchowk	981314123	16	

a. Insert a record on the table.
b. Display all the record of the table .
c. Display record whose Address is Itahari.
d. Display Name of student whose Age greater than 25.
e. Update and set age to 20 of student whose Id is 5.
f. Delete record of student whose Id is 25.
g. Delete a table.

- Create table Student_info(Id int , Name varchar(250), Address varchar(250),Phoneno varchar(250), Age int , primary key (Id));
- Insert into Student_info(id,name,address,age) values(3,”Ramkumar”, “itahari”,25);
- Select * from Student_info ;
- Select * from Student_info where Address=“itahari”
- Select name from Student_info where Age >25;
- Update Student_info set age=20 where Id=25;
- Delete from Student_info where Id=25;
- Drop table Student_info;

AND OR NOT

- ```
  SELECT * FROM Customers
  WHERE Country='Germany' AND City='Berlin’;
  ```
- ```
  SELECT * FROM Customers
  WHERE City='Berlin' OR City='München';
  ```
- ```
  SELECT * FROM Customers
  WHERE Country='Germany' OR Country='Spain’;
  ```
- ```
  SELECT * FROM Customers
  WHERE NOT Country='Germany';
  ```

Order BY

- `SELECT * FROM Customers ORDER BY Country;`
- `SELECT * FROM Customers ORDER BY Country DESC;`
- `SELECT * FROM Customers ORDER BY Country ASC`

Null Value

- A field with a NULL value is a field with no value.
- If a field in a table is optional, it is possible to insert a new record or update a record without adding a value to this field. Then, the field will be saved with a NULL value.
-
```
SELECT CustomerName, ContactName, Address
FROM Customers WHERE Address IS NULL;
```

- `SELECT MAX(Price) AS LargestPrice FROM Products;`
- `SELECT DISTINCT Country FROM Customers;`
- `SELECT COUNT(DISTINCT Country) FROM Customers;`
- `SELECT COUNT(ProductID)FROM Products;`
- `SELECT AVG(Price) FROM Products;`
- `SELECT SUM(Quantity) FROM OrderDetails;`
- `SELECT * FROM CustomersWHERE CustomerName LIKE 'a%’;`
- `SELECT * FROM Customers WHERE CustomerName LIKE '%a’;`
- `SELECT * FROM Customers WHERE CustomerName LIKE '%or%’;`
- `SELECT * FROM Products WHERE Price BETWEEN 10 AND 20;`
- `SELECT * FROM Customers WHERE Country IN ('Germany', 'France', 'UK');`

Products

ProductID	ProductName	SupplierID	CategoryID	Unit	Price
1	Chais	1	1	10 boxes x 20 bags	18
2	Chang	1	1	24 - 12 oz bottles	19
3	Aniseed Syrup	1	2	12 - 550 ml bottles	10
4	Chef Anton's Cajun Seasoning	1	2	48 - 6 oz jars	22
5	Chef Anton's Gumbo Mix	1	2	36 boxes	21.35