**SQL**

1. Database is use to store the data in the specific format.
2. These data will be store in a secondary memory so, it can be used at any point of time.
3. Data can be replicated.
4. Large amount data can be handle using Database.
5. To interact with Data you can use query language.
6. Different Database available
   1. **Relational Database**
      1. It is use to store the data in the structure format.
      2. That is the data will be store in the form of table (row and column).
      3. Data can be distributes into multiple tables.
      4. The multiple tables can be related with each other with foreign key and primary relation.
      5. Example: **MySql**, Oracle, Sql Server, Postgrace, H2, IBM, DB2 etc..
   2. Document Database
      1. The data store in the form of JSON document.
      2. Example: MongoDB, Cassendra etc.
   3. Graph Database
      1. In Graph database the data will be store in any form but it will be represented as a graph format.
      2. Example: Neo4J etc.

Database Download and installation

MySql:

<https://dev.mysql.com/downloads/installer/>



Install MySql:

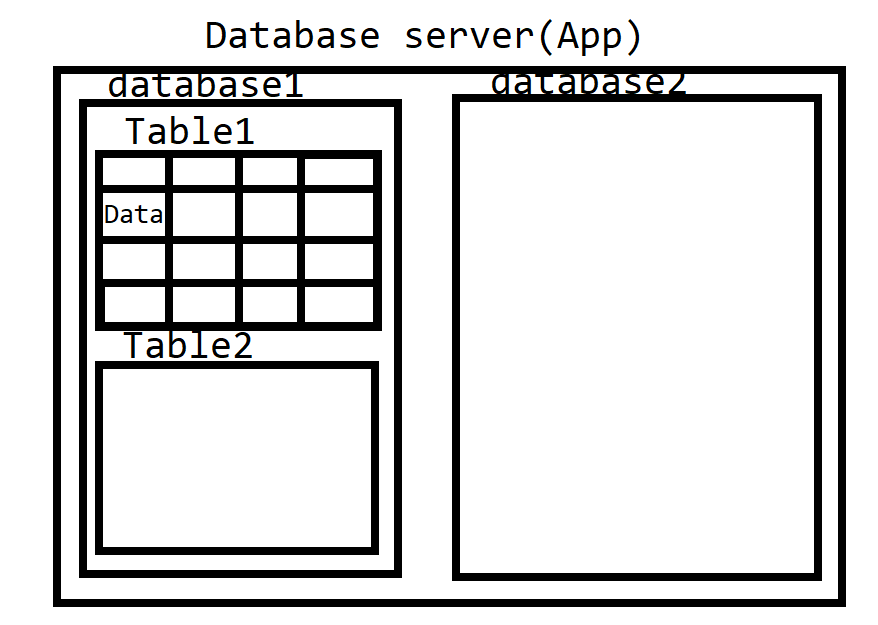
<https://www.youtube.com/watch?v=OM4aZJW_Ojs>

**Important**

Note down the port number (3306), username (root) and password during the Setup.

**MySql Jar File**

<https://repo1.maven.org/maven2/com/mysql/mysql-connector-j/8.0.32/mysql-connector-j-8.0.32.jar>



**SQL Queries**

1. SQL is a query language which is use to interact with the database.
2. SQL is divided into multiple categories
   1. DDL
      1. DDL stands for Data Definition Language
      2. This is use to **create**, **alter** (modify), **drop** (remove) the database objects.
      3. Table, database, index, procedures etc. are called as database objects
   2. DML
      1. DML stands for Data Manipulation Language.
      2. Is use to work with the data from the table.
      3. **Insert, update, delete** the records/data form the table
   3. DCL
      1. DCL stands for Data Control Language
      2. To Add or remove the restrictions on the database operations.
      3. You can set the user specific access/restriction.
      4. **GRANT** and **REVOKE** are the operations.
   4. TCL
      1. TCL stands for Transaction Control Language.
      2. You can create group of query, which can be added inside a single transaction and that transaction can be implemented completely or reverted all the changes.
      3. This can be manage by **COMMIT, ROLLBACK and SAVEPOINT**
   5. DQL
      1. DQL stands for Data Query Language
      2. Is use to retrieve the data form the database.
      3. **SELECT** type of query is use here.

**Data Type in MySql**

1. There are 3 types of data type in SQL
   1. String Data Type
   2. Numeric Data Type
   3. Date Time Data Type
2. <https://www.w3schools.com/mysql/mysql_datatypes.asp>

**DDL Query**

**Create and Use Database**

1. Create database

CREATE DATABASE <NAME>;

Example: CREATE DATABASE fsd01march;

1. Use Database

USE <Name>

Example: USE fsd01march;

**Create Table**

Syntax:

CREATE TABLE <name> (columnname DataType(size), columnname DataType(size), …. );

Example:

CREATE TABLE employee(

id int,

name varchar(20),

salary double,

gender char(1),

doj date,

status Boolean

);

**Describe the table**

To Get the structure of the table.

Syntax: desc <table\_Name>;

Example: desc employee;

**Alter Table**

1. Alter is use to modify the database objects such as table.
2. Alter query can be use by 4 ways
   1. Add New column
      1. Syntax: ALTER TABLE <tablename> **ADD COLUMN** <columnname> <datatype>
      2. Example: ALTER TABLE employee ADD COLUMN email varchar(50);
   2. Modify existing column
      1. You can modify the data type, length or default value etc.
      2. Syntax: ALTER TABLE <tablename> **MODIFY COLUMN** <columnname> <datatype>
      3. Example: ALTER TABLE employee MODIFY COLUMN name varchar(30);
   3. Remove the existing column
      1. Syntax: ALTER TABLE <tablename> **DROP COLUMN** <columnname>
      2. Example: ALTER TABLE employee DROP COLUMN status;
   4. Rename the column
      1. Syntax: ALTER TABLE <tablename> **RENAME COLUMN** <old\_columnname> TO <new\_columnname>
      2. Example: ALTER TABLE employee RENAME COLUMN doj TO joiningdate;

**Drop database objects**

1. Using drop query you can delete the database objects such as database, table, index, view etc.
2. This will delete the object permanently.
3. Syntax: DROP <ObjectType> <name>;
4. Example: DROP TABLE employee

DROP DATABASE fsd04march;

**DML**

**Insert Record**

Syntax:

**INSERT INTO <table\_name>(column1, column2) VALUES(value1, ‘Value1’,., ., ., );**

Example:

INSERT INTO employee(id,name,salary,gender,joiningdate,email) VALUES(1,'Abc',234223.4,'M','2020-02-22','abc@g.com');

INSERT INTO employee VALUES(2,'Xyz',214223.4,'F','2020-02-12','xyz@g.com');

INSERT INTO employee VALUES(4,'Test1',14223.4,'M','2021-02-12','t1@g.com'),

(5,'Test2',42223.4,'F','2021-03-22','t2@g.com'),

(6,'Test3',65223.4,'F','2022-04-02','t3@g.com'),

(7,'Test4',41223.4,'M','2018-01-11','t4@g.com');

INSERT INTO emp\_backup SELECT \* from employee;

**Update Record**

Syntax:

**UPDATE <table\_name> SET column=’Value’, column=’Value’ clause**

Example:

UPDATE employee SET joiningdate='2019-11-19' WHERE id=3;

UPDATE employee SET salary=salary + ((salary\*5)/100);

UPDATE employee SET salary=salary + ((salary\*5)/100) WHERE gender='F';

**Delete Record**

**Syntax:**

DELETE FROM <table\_name> clause;

Example:

DELETE FROM employee WHERE id=3;

DELETE FROM employee WHERE gender='M';

DELETE FROM employee;

**DQL**

1. In DQL select type of queries can be use.
2. You can select a rows and columns using this query.
3. To select a specific rows then you have to use the clauses.
4. DQL queries are use to retrieve a data in different format.
5. Syntax:

**SELECT column\_name/\* FROM <table\_name> clause**

Example:

**SELECT \* FROM employee;**

**SELECT name, email, joiningdate FROM employee;**

**SELECT name, email, joiningdate AS doj FROM employee;**

**SELECT \* FROM employee WHERE gender='M';**

**Clause**

1. Clauses are use to restrict the number of records getting affected or selected.
2. Clauses are a conditions which are applied on rows/data.
3. Using this you can limit the data to be handle.
4. Different type of clauses
   1. Where
      1. In this clause you can select a specific rows by setting a condition(s).
      2. This condition can be for equal values or with the relations operators.
      3. There can be a more than one condition which can be combine together by AND, OR clause
      4. Example

**SELECT \* FROM employee WHERE salary>60000 AND gender='F' ;**

**SELECT \* FROM employee WHERE gender='M';**

**SELECT \* FROM employee WHERE salary>60000 OR gender='F' ;**

**SELECT \* FROM employee WHERE salary>50000 AND salary<100000;**

* 1. Like
     1. Is use to select the details based on partial data.
     2. Like clause has to use with Where clause only.
     3. Syntax:

SELECT \* FROM <table\_name> WHERE name LIKE ‘%value%’

* + 1. In the above query percent(%) denotes any values.
    2. Example:

**SELECT \* FROM employee where name LIKE 'Tes%';**

**SELECT \* FROM employee where name LIKE '%Tes%';**

**SELECT \* FROM employee where name LIKE '%Tes’;**

* 1. IN
     1. This clause is use to select the records based on the list of values.
     2. This is exactly same as the OR clause.
     3. You can provide a list of values which needs to be compare with the actual values.
     4. Syntax:

SELECT \* FORM <table\_name> WHERE <column\_name> **IN (value1, value2)**

* + 1. Example:

select \* from employee WHERE year(joiningdate) IN (2018, 2020, 2021);

* 1. Between
     1. This Clause is use to select the records based on range.
     2. This is same as AND clause.
     3. Syntax:

SELECT \* FROM <table> WHERE <column> **BETWEEN value1 AND value2**;

* + 1. Example:

**SELECT \* FROM employee WHERE salary BETWEEN 10000 AND 50000;**

* 1. Limit
     1. It is use to limit the number of records to be selected.
     2. Limit can be use with the other clause.
     3. Syntax:

SELECT \* FROM <table> **LIMIT <number>;**

Exmample: select \* from employee **LIMIT 4;**

* 1. Order by clause
     1. This clause is use to select the records into a specific order like ascending or descending order.
     2. By Default order by clause selects the records in Ascending order.
     3. You needs to provide the column(s) name which needs to be consider for order.
     4. Syntax:

SELECT \* FROM <table> **ORDER BY <column> ASC/DESC**;

Example:

**select \* from employee ORDER BY salary ASC;**

**select \* from employee ORDER BY salary DESC;**

**select \* from employee ORDER BY salary,name ASC;**

* 1. Group by
     1. It is use to group the records and perform the operation on the group records.
     2. The operations can be an aggregate functions such as count, sum, min, max, avg.
     3. **Syntax:**

**SELECT <function(column)> FROM EMPLOYEE GROUP BY <column\_name>**

Example:

select gender, count(\*) from employee GROUP BY gender; // Count by Gender

select count(\*) from employee GROUP BY year(joiningdate); // count by joining year

select max(salary) from employee GROUP BY gender; // max salary by gender

* 1. Having
     1. Is use to apply condition on the group records.
     2. It can be use only with group by clause and must be present at the end of the query.
     3. Syntax:

**SELECT <function(column)> FROM EMPLOYEE GROUP BY <column\_name> HAVING <condition>**

Example:

select year(joiningdate) AS year, count(\*) AS empcount from employee **GROUP BY year(joiningdate) HAVING empcount>1;**

SELECT year(joiningdate) as year, max(salary) as salary FROM employee **GROUP BY year(joiningdate) HAVING salary>50000;**

**Constraints:**

1. It is use to set the rules on the column.
2. These rules will be applied whenever you are creating a records.
3. There are different contains available
   1. Unique
      1. It is use to maintain the uniqueness in the values.
   2. Not Null
      1. It is use to restrict the null values for the column.
   3. Check
      1. You can set the condition on the values to be inserted for a column.
   4. Default
      1. Default is use to set the default value for a column which will inserted if no user specific value provided.
   5. Primary Key
      1. Is use to make the column value unique and not null
   6. Foreign Key
      1. Is use to make the reference of one table into another. With this you can establish the relation between 2 table. Foreign Key can be duplicate and can be null also. The primary of one table only can be use as a foreign key into another table.

Student Table

Id – Unique and not null

Name – not null

Contact - unique

Age – must be between 3-60

Gender – Must be M, F, O

City – Default ‘Pune’

**CREATE TABLE student(**

**id int primary key,**

**name varchar(30) NOT NULL,**

**contact varchar(10) UNIQUE,**

**city varchar(20) DEFAULT 'Pune',**

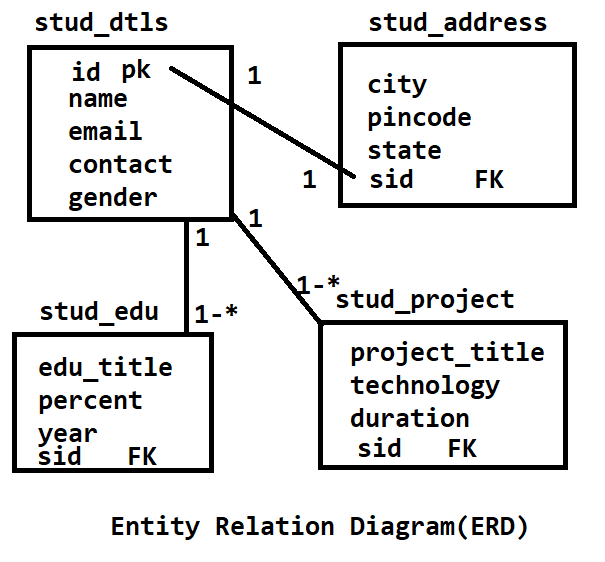
**age int,**

**gender char(1),**

**CONSTRAINT chk\_age CHECK (age BETWEEN 3 AND 60),**

**CONSTRAINT chk\_gender CHECK (gender IN (,'M', 'F' 'O'))**

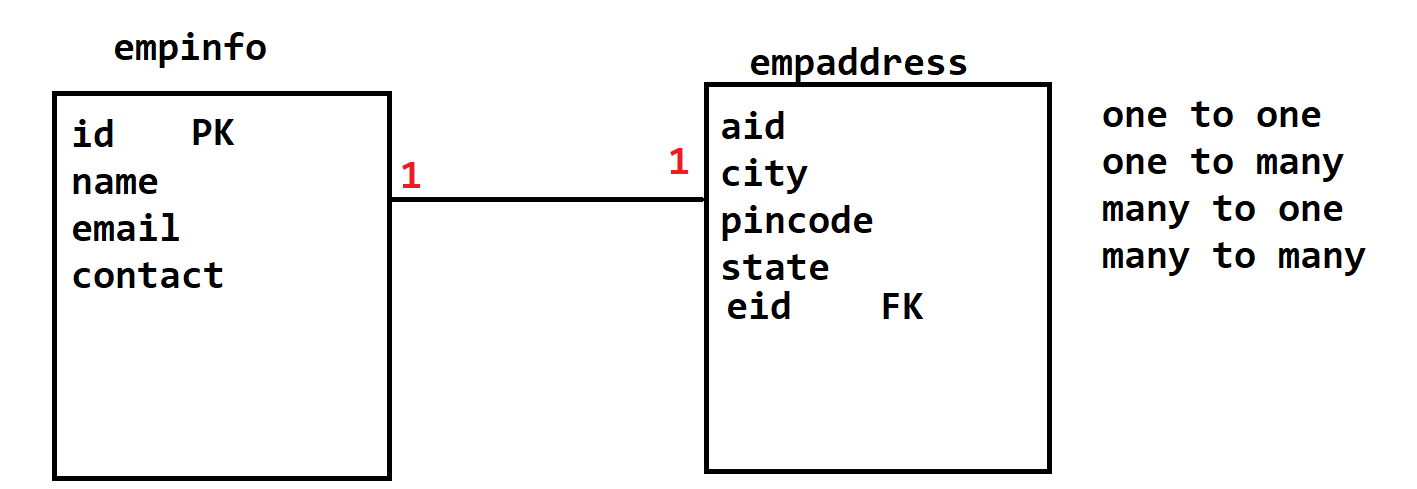
**);**

****

**Joins**

1. Join is use to combine multiple tables to get the result.
2. Join is use in the select query only.
3. To Join the table you must have to set the relation between a table using primary key and foreign key.
4. Join are of different types.
   1. Cross Join
   2. Inner join
   3. Outer Join
      1. Left Outer Join
      2. Right Outer Join
      3. Full join





create table empinfo(

id int primary key auto\_increment,

name varchar(20),

email varchar(50),

contact varchar(10)

);

create table empaddress(

aid int primary key auto\_increment,

city varchar(20),

pincode varchar(8),

state varchar(10),

eid int,

FOREIGN KEY(eid) REFERENCES empinfo(id)

);

**Cross Join**

1. Is use to map the record from one table with the all records from another table.
2. Cross Join used without any clause
3. Example:

SELECT name, email,city, pincode FROM empinfo **CROSS JOIN** empaddress;

**Inner Join**

1. Inner join is use to get the matching records from both the tables.
2. For getting the matching records you have to apply a clause on the Primary and foreign key.
3. Example:

SELECT name, email,city, pincode

FROM empinfo **INNER JOIN** empaddress

WHERE empinfo.id=empaddress.eid;

SELECT name, email, contact, city, pincode, state

FROM empinfo INNER JOIN empaddress

ON empinfo.id = empaddress.eid;

**Left outer Join**

1. It is use to get all the records(matching and non-matching) from left table and only matching record from right table.
2. Example:

SELECT name, email,city, pincode FROM empinfo **LEFT OUTER JOIN** empaddress ON empinfo.id=empaddress.eid;

SELECT name, email, contact, city, pincode, state

FROM empinfo LEFT OUTER JOIN empaddress

ON empinfo.id = empaddress.eid;

**Right outer Join**

1. It is use to get all the records(matching and non-matching) from right table and only matching record from left table.
2. Example:

SELECT name, email,city, pincode FROM empinfo **RIGHT OUTER JOIN** empaddress ON empinfo.id=empaddress.eid;

SELECT name, email, contact, city, pincode, state

FROM empinfo LEFT OUTER JOIN empaddress

ON empinfo.id = empaddress.eid;

**Full outer Join**

1. It is use to get all the records(matching and non-matching) from both the tables.
2. Example:

SELECT name, email,city, pincode FROM empinfo **FULL OUTER JOIN** empaddress ON empinfo.id=empaddress.eid;

mysql> create table empcompnay(

-> cid int primary key auto\_increment,

-> name varchar(20),

-> loaction varchar(50),

-> eid int,

-> FOREIGN KEY(eid) REFERENCES empinfo(id)

-> );

mysql> SELECT e.name, e.email,a.city, a.pincode, c.name AS Company, c.loaction

-> FROM empinfo e INNER JOIN empaddress a

-> ON e.id=a.eid

-> INNER JOIN empcompnay c

-> ON e.id=c.eid;

mysql> SELECT e.name, e.email,a.city, a.pincode, c.name AS Company, c.loaction

-> FROM empinfo e INNER JOIN empaddress a

-> ON e.id=a.eid

-> LEFT OUTER JOIN empcompnay c

-> ON e.id=c.eid;