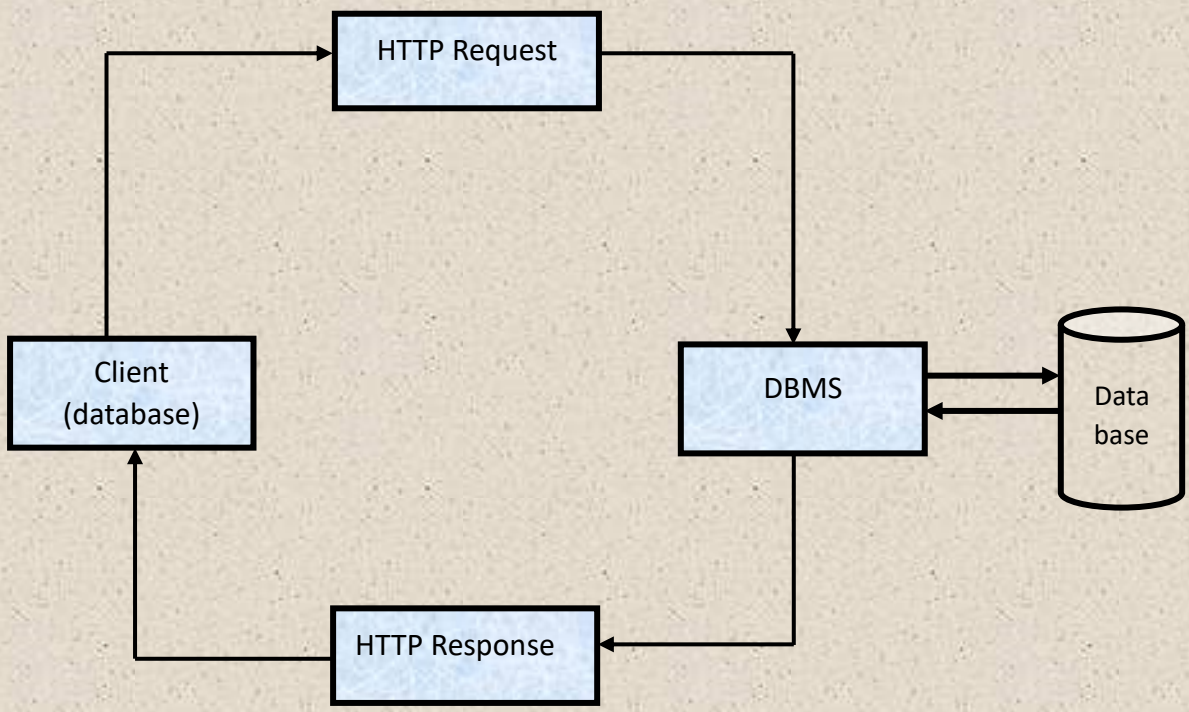
DBMS.

- **DBMS :-** Database Management System. A software application used to manage our DB is called DBMS. It is a software application which is used to store all the user related information.
  1. Data is highly secured.
  2. Data is stored in an organized manner.
  3. Data is easily fetched.
- **Database** :- database is a collection of meaning full data in a format that can be easily accessed digital.
- **DBMS ek server ki tarah act karta hai.**
- **Server.**
- It helps in storing and managing user data.
- **Client server communication.**
  1. step: the client needs to store or retrieve user information.
  2. step: the client finds a relevant server that provides the service.
  3. step: the client get close to the server to communicate with each other.
  4. step: the client will send the request to the server.
  5. step: the server will fetch and processes the client's request.
  6. step: the server will sends a response to the client.
- **Client server communication diagram.**



- **Types of Data base.**
  1. Relational data base.
- **Relational data base :** data stored in tables.
- **Example of Relational data base.**
- MySQL, ORACLE, SQL Server.
  2. Non-Relational data base.
- **Non-Relational data base :** data not store in tables.
- **Example of Non- Relational data base.**
- Mongo DB.
- **SQL Data types.**

DATATYPE	DESCRIPTION	USAGE
CHAR	string(0-255), can store characters of fixed length	CHAR(50)
VARCHAR	string(0-255), can store characters up to given length	VARCHAR(50)
BLOB	string(0-65535), can store binary large object	BLOB(1000)
INT	integer(-2,147,483,648 to 2,147,483,647)	INT
TINYINT	integer(-128 to 127)	TINYINT
BIGINT	integer(-9,223,372,036,854,775,808 to 9,223,372,036,854,775,807)	BIGINT
BIT	can store x-bit values. x can range from 1 to 64	BIT(2)
FLOAT	Decimal number - with precision to 23 digits	FLOAT
DOUBLE	Decimal number - with 24 to 53 digits	DOUBLE
BOOLEAN	Boolean values 0 or 1	BOOLEAN
DATE	date in format of YYYY-MM-DD ranging from 1000-01-01 to 9999-12-31	DATE
TIME	HH:MM:SS	TIME
YEAR	year in 4 digits format ranging from 1901 to 2155	YEAR

- **DBMS se hum communication karne ke liya hamen specific language SQL queries.**
- **SQL.**
- SQL :- SQL is Structure Query Language.
- It is a standard interactive programming language used to communicate with the DBMS.
- SQL is a programming language used to interact with relational database.
- It is used to perform CRUD operations.
  1. **Create :** Adds or inserts rows into a table.
  2. **Read :** Selects (retrieves) rows from a table.
  3. **Update :** Modifies rows in a table.
  4. **Delete :** Removes rows from a table.
- **Question : are SQL and MySQL the same thing.**
- SQL : it is a language.
- MySQL : it is a DBMS.
- **Database structure.**
- Inter related database.
- **What is table.**
- Table is a collection of rows and column.
- Column direction vertically
- Rows direction horizontally.
- **Creating our first database.**
  1. CREATE DATABASE db. _ name.
  2. DROP DATABASE db. _name.
- **Creating our first table.**
- **Syntax:**
- **USE DATABASE NAME;**
- **CREATE TABLE table _ name;**
  Column _ name1 datatype constraint,
  Column _ name2 datatype constraint,
  Column _ name3 datatype constraint,);

- **CREATE TABLE Students**(id INT PRIMARY KEY, Name VARCHAR(50),Age INT NOT NULL);
- **INSERT INTO Students**
  Values(1,'Shaan',20);
  Values(2,'Pawan',18);
  Values(3,'Shradha',24);
  Values(4,'Ujjwal',19);
- **SELECT * FROM college; print karna ka liya.**
- **SQL Data types.**

1. **Signed data type.**

- Variable with signed number can store 0, positive and negative number. Such as.
  1. Int
  2. Float.
  3. Double.
  4. TINYINT.

2. **Unsigned data type.**

- Variables with unsigned number can store only 0 and positive numbers. Example. Age and salary.
- TINYINT (0 to 255). After change in unsigned.
- **Types of SQL Commands.**
  1. DDL: Data definition language. Create, alter, rename, truncate and drop.
  2. DQL: Data Query Language. Select.
  3. DML: Data Manipulation Language. Insert, update and delete.
  4. DCL: Data Control Language. Grant and revoke permissions to users.
  5. TCL: Transaction Control Language. Start transactions, commit, rollback
- **Data Base related queries.**
- CREATE DATABASE data base _name;
- CREATE DATABASE IF NOT EXISTS data base _ name;
- CREATE DATABASE IF NOT EXISTS college;
- DROP DATABASE data base _ name;
- DROP DATABASE IF EXISTS data base _ name;
- DROP DATABASE IF EXISTS shaan;
- SHOW DATABASE;
- SHOW TABLE;

Keys

- **Keys:** An SQL key, in Database Management Systems, is either a single attribute (or column) or a set of attributes that can uniquely identify rows (or tuples) in a table. SQL keys ensure that there are no rows with duplicate values.
- **There are two types of keys.**
  1. Primary key.
  2. Foreign key.
- **Primary key.**
- It is a column (or set of columns) in a table that uniquely identifies each row. (a unique id).
- There is only 1 PK & it should be NOT NULL.
- **Example:**

ID	City _name
1	Puna
2	Mumbai
3	Delhi
4	Dehradun

- **Foreign key.**
- A foreign key is a column (or set of columns) in a table that refers to the primary key.
- There can be multiple FKs.
- FKs can have duplicate & null values.
- **Example:**

City_ id	City _name
1	dehradun
2	Mumbai
3	Delhi
1	Dehradun

Constraints

- **First Constraints PRIMARY KEY.**
- **SQL constraints are used to specify rules for data in a table.**
- **NOT NULL:** columns cannot have a null value. Col1 int NOT NULL.
- **UNIQUE:** all values in column are different. Col2 int UNIQUE.
- **PRIMARY KEY:** makes a column unique & not null but used only for one. Id in primary key.
- **How to make primary key.**
- **CREATE TABLE Students**(id INT PRIMARY KEY, Name VARCHAR(50),Age INT NOT NULL);
- **Second example.**
- **CREATE TABLE Students**(id INT, Name VARCHAR(50),Age INT, City VARCHAR(20) PRIMARY KEY(id, name);
  // every student name and id together primary key.
- **Second Constraints FOREIGN KEY.**
- **FOREIGN KEY:** prevent actions that would destroy links between tables.
- CREATE table temp (Cust _id INT, FOREIGN KEY (Cust_ id) reference Customer (id));
- **DEFAULT:** sets the default value of a column.
- Salary INT DEFAULT 25000.
- **Check Constraints.**
- **Check:** it can limit the values allowed in a column.
- **CREATE TABLE City**(id INT PRIMARY KEY, City VARCHAR(50), age INT, CONSTRAINTS age_ check CHECK (age >= 18 AND City =”Delhi”));
- **CREATE TABLE NewTab**(age INT CHECK (age>= 18));

SELECT IN DETAIL

- **SELECT:** It is used to access the records from one or more database tables and views. used to select any data from the database.
- **Basic Syntax.**
- SELECT col1, col2 FROM table_ name;
- SELECT Name, Marks FROM Students;
- **DISTINCT special keyword.**
  // DISTINCT meaning is unique show the unique values.
- SELECT DISTINCT City FROM Students;
- **To select all**
- SELECT * FROM table_ name;
- SELECT * FROM Students;

Where Clause

- **Where Clause:** specifies criteria that field values must meet for the records that contain the values to be included in the query results. to define some conditions.
- **Syntax.**
- SELECT col1, col2 FROM table_ name;
- **Where conditions;**
- SELECT * FROM Students WHERE Marks > 80;
- SELECT * FROM Students WHERE City = 'Mumbai';
- SELECT * FROM Students WHERE Marks > 80 AND City = 'Mumbai';

- **Where Clause.**
- **Using operators in WHERE.**
- **Arithmetic operators.**
- Addition = (+). Subtraction = (-). Multiplication = (*).
- Divide = (/). Modulo = (%). Exponential = (**).
- **Arithmetic operators queries syntax.**
- SELECT * FROM Class_Students WHERE Marks > 80 AND City = 'Mumbai';
- **Comparison operators.**
- Equal to = (==). Not equal to = (!=). Less than = (<).
- Less equal to = (<=). Greater then = (>). Greater then equal to = (>=)
- **Comparison operators queries syntax.**
- SELECT * FROM Class_Students WHERE Marks = 93;
- SELECT * FROM Class_Students WHERE Marks > 90;
- **Logical operators.**
- **And = &&. (to check for both conditions to be true) queries syntax.**
- SELECT * FROM Class_Students WHERE Marks > 80 AND City = 'Mumbai';
- **Or = ||.(to check for both conditions to be true) queries syntax.**
- SELECT * FROM Class_Students WHERE Marks > 90 OR City = 'Mumbai';
- **Between (selects for a given range) queries syntax.**
- SELECT * FROM Class_Students WHERE Marks > 90 OR City = 'Mumbai';
- **In operators (matches any value in the list) queries syntax.**
- SELECT * FROM Class_Students WHERE City IN ('Delhi','Mumbai');
- **Not =! Operators (to negate the given condition) queries syntax.**
- SELECT * FROM Class_Students WHERE City NOT IN ('Delhi','Mumbai');
- **Limit Clause.**
- Set an upper limit on number of (tuples) rows to be returned.
- SELECT * FROM Class_Students LIMIT 3;
- SELECT * FROM Class_Students WHERE Marks > 80 LIMIT 3;

- **Order By Clause.**
- To sort in ascending (ASC) and descending order (DESC).
- Ascending order = 1 2 3 4 5.
- Descending order = 5 4 3 2 1.
- SELECT * FROM Class_Students ORDER BY City ASC;
- SELECT * FROM Class_Students ORDER BY City DESC;
- SELECT * FROM Class_Students ORDER BY City DESC LIMIT 3;

Aggregate Function

- **Aggregate function:** Aggregate function perform a calculation on a set of values, and return a single value.
- **Most commonly use function.**
  1. **COUNT() =** SELECT COUNT(Marks) FROM Students;
  2. **MAX() =** SELECT MAX (Marks) FROM Students;
  3. **MIN() =** SELECT MIN (Marks) FROM Students;
  4. **SUM() =** SELECT SUM (Marks) FROM Students;
  5. **AVG() =** SELECT AVG (Marks) FROM Students;

Group By Clause

- **Group By Clause**: Group rows that have the same values into summary rows.
- It collects data from multiple records and group the result by one or more column.
- Generally, we use group by with some **aggregate function**.
- **Group making in city.**
- SELECT City FROM Class_Students GROUP BY City;
- **Count number of students in each city.**
- SELECT City, COUNT(Rollno)FROM Class_Students GROUP BY City;
- **Group of bases of name.**
- SELECT City, Name, COUNT(Rollno)FROM Class_Students GROUP BY City, Name;
- **All Students average marks of city.**
- SELECT City, Avg(Marks)FROM Class_Students GROUP BY City;

Having Clause.

- **Having Clause.**
- Similar to Where. i.e. applies some condition on rows.
- Used when want to apply any conditions after grouping.
- **Count number of students in each city where max marks cross 90.**
- SELECT City, Count(rollno) FROM Students GROUP BY City Having MAX(marks) > 90;
- **Condition use.**
- Where condition use only all rows.
- Having condition use only all column groups.
- **How many students grade A which city in.**
- SELECT City FROM Students WHERE grade = "A" GROUP BY City HAVING Max(Marks) > 90 ORDER BY City ASC;

Revisiting FK

- My SQL Visual eyes / visualizes:
- Step 1: My SQL Data base option and jo reverse engineer option click.
- Step 2: continue and continue select the database as college etc.

Cascading for FK

- **Cascading for UK.**
- **On Delete Cascade.**
- When we create a foreign key using this option, it deletes the referencing rows in the child table when the referenced row is deleted in the parent table which has a primary key.
- **On update cascade.**
- When we create a foreign key using UPDATE CASCADE the referencing rows are updated in the child table when the referenced row is updated in the parent table which has a primary key.

- **Code of cascading.**

```
USE College;
CREATE TABLE Course(
Course_id INT PRIMARY KEY,
Course_name VARCHAR(50));

INSERT INTO Course
VALUES
(101,'English'),
(102,'Hindi'),(103,'Maths'),
(104,'Science')
,(105,'IT');

UPDATE Course SET Course_id = 106  WHERE Course_id = 105;

SELECT *FROM Course;

CREATE TABLE Teacher(
Teacher_id INT PRIMARY KEY,
Teacher_name VARCHAR(50),
Course_id INT,
FOREIGN KEY (Course_id) REFERENCES Course(Course_id)
ON UPDATE CASCADE
ON DELETE CASCADE);


INSERT INTO Teacher
Values
(101,'Anil',101),
(102,'Bhunika',102),
(103,'Chetan',103),
(104,'Dhruv',104),
(105,'Emanuel',105);

SELECT *FROM Teacher;
```

Table Related Queries

- **Table Related Queries.**
- **Update** (to update existing rows).
- **Delete** (to delete existing rows).
- **Update queries.**
- UPDATE table_name
  SET Col1 = Val1, Col2 = Val2
  WHERE Condition;
- UPDATE Students SET grade = "o" WHERE grade = "A";
- UPDATE Students SET Marks = 82 WHERE Rollno = 105;
- UPDATE Students SET grade = "B" WHERE Marks BETWEEN 80 AND 90;
- UPDATE Students SET Marks = Marks + 1; // update one number.
- **Error: you are using safe update mode on.**
- SET SQL_SAFE_UPDATES = 0;
- **Delete queries.**
- DELETE FROM Table_Name WHERE condition;
- DELETE FROM Students WHERE marks < 33;
- Data Delete: DELETE FROM Students;
- **Alter Queries.**
- Alter (to Change the Schema)
- Alter Schema meaning = design, column, datatypes,constraints.
- **ADD Column**
  ALTER TABLE table_name
  ADD COLUMN column_name datatype constraint;
  ALTER TABLE Students ADD COLUMN Age INT;
  ALTER TABLE STUDENT ADD COLUMN age INT NOT NULL DEFAULT 19;
- **DROP Column**
- **DROP DELETE THE ALL TABLE.**
- ALTER TABLE table_name
  DROP COLUMN column_name;
  ALTER TABLE Students DROP COLUMN Age;
- **RENAME Table**
- ALTER TABLE table_name
  RENAME TO new_table_name;
  ALTER TABLE Students RENAME TO STUDENT;

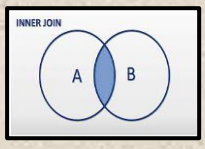
- **CHANGE COLUMN**
- CHANGE Column (rename)
  ALTER TABLE table_name
  CHANGE COLUMN old_name new _name new_datatype new_constraints.
  ALTER TABLE STUDENT RENAME COLUMN Rollno TO Student_Rollon;
- **MODIFY COLUMN**
- MODIFY Column (modify datatype/ constraint)
  ALTER TABLE table_name
  MODIFY col_name new_datatype new_constraint;

TRUNCATE

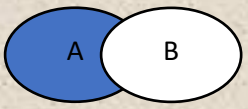
- **TRUNCATE DELETE ALL DATA.**
- **TRUNCATE**(to delete table's data).
- TRUNCATE TABLE TABLE_NAME.
- TRUNCATE TABLE Students;

Join

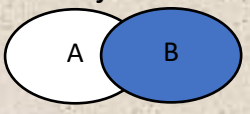
- **Join.**
- Join is used to combine rows from two or more tables, based on a related column between them.
- **Types of join.**



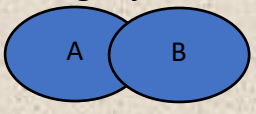
1. Inner join.



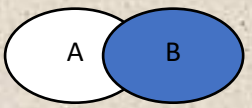
2. Left join.



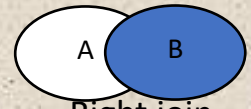
3. Right join.



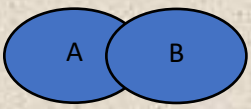
4. Full join.



Left join



Right join



Full join

5.Outer join.

- **Inner Join.**
- Returns records that have matching values in both tables.
- **Syntax.**

  SELECT column(s)
  FROM table A
  INNER JOIN table B
  ON table A. col_name = Table B. col_name;
  USE alias = alternate name.
- **Syntax code.**

  SELECT *FROM STUDENT_A
  INNER JOIN STUDENT_B
  ON STUDENT_A.Student_id = STUDENT_B.Student_id;

- **Left Join.**
- Returns all records from the left table, and the matched records from the right table.
- **Syntax.**

  SELECT column(s)
  FROM tableA
  LEFT JOIN tableB
  ON tableA.col_name = tableB.col_name.
- **Syntax code.**

  SELECT *FROM STUDENT_A
  LEFT JOIN STUDENT_B
  ON STUDENT_A.Student_id = STUDENT_B.student_id;
- **Left exclusive join.**

  SELECT *FROM STUDENT_A
  LEFT JOIN STUDENT_B
  ON STUDENT_A.Student_id = STUDENT_B.student_id
  WHERE STUDENT_B.Student_id IS NULL;
- **Right Join.**
- **Returns all records from the right table, and the matched records from the left table.**
- **Syntax.**

  SELECT column(s)
  FROM tableA
  RIGHT JOIN tableB
  ON tableA.col_name = tableB.col_name.
- **Syntax code.**

  SELECT *FROM STUDENT_A
  RIGHT JOIN STUDENT_B
  ON STUDENT_A.Student_id = STUDENT_B.student_id;
- **Right exclusive join.**

  SELECT *FROM STUDENT_A
  RIGHT JOIN STUDENT_B
  ON STUDENT_A.Student_id = STUDENT_B.student_id
  WHERE STUDENT_A.Student_id IS NULL;

- **Full Join.**
- Returns all records when there is a match in either left or right table.
- **Syntax.**

```
SELECT column(s)
FROM tableA
LEFT JOIN tableB
ON tableA.col_name = tableB.col_name.
UNION
SELECT  column(s)
FROM tableA
RIGHT JOIN tableB
ON tableA.col_name = tableB.col_name.
```

- **Syntax code.**

```
SELECT *FROM STUDENT_A
LEFT JOIN  STUDENT_B
ON STUDENT_A.Student_id = STUDENT_B.student_id
UNION
SELECT *FROM STUDENT_A
RIGHT JOIN STUDENT_B
ON STUDENT_A.Student_id = STUDENT_B.student_id;
```

- **Self Join**.
- It is a regular join but the table is joined with itself.
- **Syntax.**

```
SELECT column(s)
FROM table as a
JOIN table as b
ON a.col_name = b.col_name;
```

- **Syntax code.**
-

```
SELECT a.name, b.name
FROM EMPLOYEE as a
JOIN EMPLOYEE as b
ON a.id = b.manger_id;
```

- **UNION.**
- It is used to combine the result-set to two or more SELECT statements. Given UNIQUE records.
- To use it.
- Every SELECT should have same same no.of columns.
- Columns must have similar data types.
- Columns in every SELECT should be in same order.
- **Syntax.**
  SELECT column(s) FROM tableA
  UNION
  SELECT column(s) FROM tableB
- **Syntax code.**
  SELECT Student_name FROM STUDENT
  UNION
  SELECT Student_name FROM Studentb;
- **Syntax duplicate values.**
  SELECT Student_name FROM STUDENT
  UNION ALL
  SELECT Student_name FROM Studentb;

SQL Sub Queries

- **SQL Sub Queries.**
- A Subquery or inner query or a Nested query is a query within another SQL queries.
- It involves 2 select statements.
- **Syntax.**
  SELECT column(s)
  FROM table_name
  WHERE col_name operator
  (subquery);
- **SQL Sub Queries.**
- Get names of all students who scored more than class average.
- **Find the avg of class.**
- SELECT AVG(Marks) FROM Students;
- **Find the names of students with marks > avg.**
- SELECT Name, Marks FROM Students WHERE Marks > 74.33333333333333;
- **New student left students number grater than and less than that is dynamic.**
- SELECT Name, Marks FROM Students WHERE Marks > (SELECT AVG(Marks) FROM students);
- **Find the even Roll-no numbers.**
- SELECT Rollno FROM Students WHERE Rollno % 2 == 0;
- **Find the name of even Roll-no numbers students.**
- SELECT Name FROM Students WHERE Rollno IN (102,104,106);
- SELECT Name,Rollno FROM Students WHERE Rollno IN (SELECT Rollno FROM Students WHERE Rollno % 2 == 0);
- **Find the students of Delhi.**
- SELECT * FROM Students WHERE City = "Delhi";
- **Find their max marks using the sublist in step 1.**
- SELECT MAX(Marks) FROM (SELECT * FROM Students WHERE City = "Delhi") as temp;
- SELECT(SELECT MAX(Marks) FROM Students), Name FROM Students;

MySQL Views

- **MySQL Views.**
- **A view is a virtual table based on the result-set of an SQL statements.**
- **View Create.**
- CREATE VIEW view1 AS SELECT Rollno, Name, Marks FROM Students;
- **View.**
- SELECT * FROM view1;