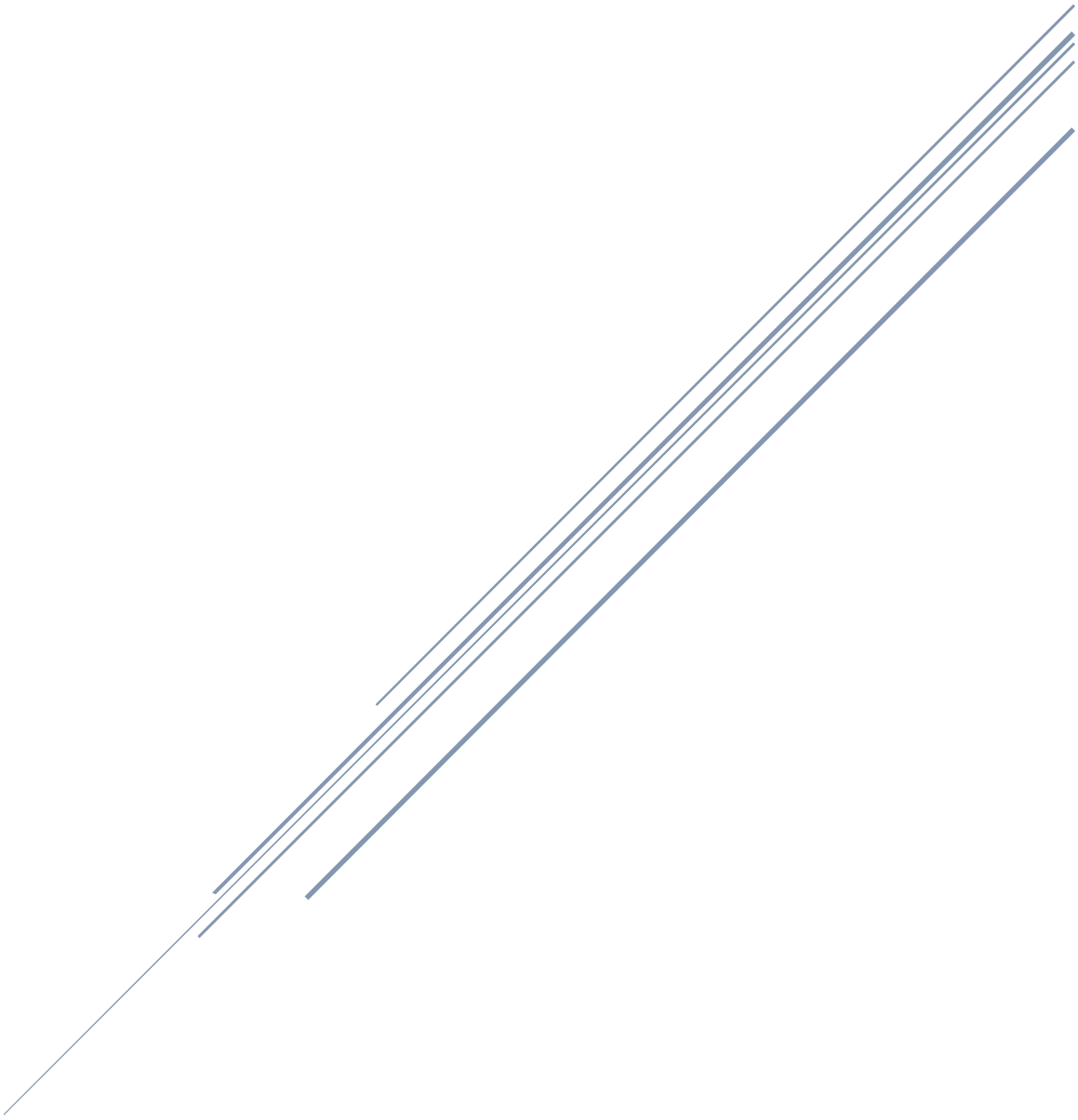
# SQL by APNA College



[School]
[Course title]

# History of data base

Before database people used to store data in flat files.

A flat file system is a type of file that stores a simple database representation. Because flat file databases lack linkages between tables.

### Why database was introduced?

There were many drawbacks of flat file system, which are given below.

### Flat file system drawbacks

- Redundancy issue means duplicate names conflict
- Security issue
- Memory issue occupying more
- Searching was difficult because of same names

# Introduction to Database

### Database

Collection of related data in a form so that I can be accessed (insert, delete,update,search) easily.

Why Databases?

- Can store large data
- Features like security, scalability etc.
- Easier to insert, update or delete data.

### DBMS (database management system)

DBMS is a software which is used to manage the database.

### There are two types of databases

1. Relational DB (SQL)
    a. Relational database (data stored in tables)
        i. Mysql, Oracle, PostgreSQL etc.

2. Non-rational Relational (NO SQL)
    a. Non relational databases(data stored in document/key-val/graphs etc)
        i. Mongodb, Cassandra, neo4j etc.

# Data base operations:

DDL (data definition language)

DQL (data query language)

DCL (data control language)

DML (data manipulation language)

# SQL(structured Query Language)

**SQL is a programming language used to interact with relational databases.**

**Table in SQL**

| | Columns | |
|---|---|---|
| rows | | |
| | | |

Mysql server > mysql workbench

**Schema:**

- **Database schema:** A database schema defines how data is organized within a relational database; this is inclusive of logical constraints such as, table names, fields,
- **Table schema:** design of table.

What are keys?

Keys are special columns in the table

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 and it should be NOT NULL.

Foreign key

A foreign key is a column ( or set of columns) in a table that refers to the primary key in another table.

FKs can have duplicate a null values.

There can be multiple FKs.

MYSQL Commands

**Database:**

CREATE DATABASE db_name; > used to create database

CREATE DATABASE IF NOT EXISTS db_name

DROP DATABASE db_name; > used to remove database

DROP DATABASE if EXISTS db_name;

SHOW DATABASES;

SHOW TABLES;

USE db_name; > used to select database

**Create Table**

CREATE TABLE table_name(

Column_name1 datatype constraint,

Column_name2 datatype constraint,

Column_name3 datatype constraint

);

INSERT into values(101,SIRAJ,18),

(102,DUJANA,20);

Table queries:

- Create
  - CREATE TABLE table_name();
    - Data types
      - Char(50)
      - Var-char(50) optimized usage of memory
      - BLOB(used to store large Objects)
      - INT
    - Constraints(Rules for data in the table)
      - NOT NULL: columns cannot have a null value.
      - Unique: all values in column are different.
      - DEFAULT: sets the default value of a column.
      - CHECK: it can limit the values allowed in a column.
      - Ex: salary INT DEFAULT 2500 or constraint age_check check(age>=18 and city=”delhi);
      - Primary Key: makes a column unique and not null used only for one.
        - Ex: create table temp ( id int, primary key(id));
        - Ex: create table temp( id int primary)
      - Foreign key: prevent actions that would destroy links between tables.
      - We can connect tables through it.
      - Primary key of another table in the table will be foreign key.

Student table

| Id | Name | T-id(foreign key) |
|---|---|---|
| 101 | siraj | 1 |

Teacher table

| Id(primary key) | name |
|---|---|
| 1 | dujana |

Foreign key (t-id) references teacher (id)

  - Ex: create table temp( cust_id int, foreign key (cust_id) references customer(id)

- Insert
  - INSERT INTO VALUES();
- Update
- Alter
- Truncate
- delete

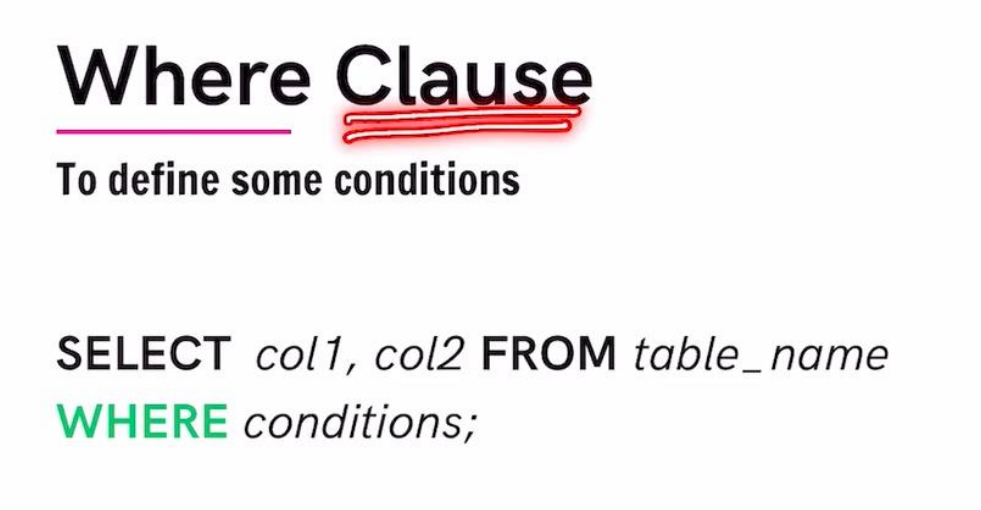
**Select Command**

Selects and show data from the DB

Syntax:

SELECT column1, column 2 FROM table_name;

SELECT * FROM table_name;



# Where Clause

## Operators

Arithmetic Operators : +(addition) , -(subtraction), *(multiplication), /(division), %(modulus)

Comparison Operators : = (equal to), != (not equal to), > , >=, <, <=

Logical Operators : AND, OR , NOT, IN, BETWEEN, ALL, LIKE, ANY

Bitwise Operators : & (Bitwise AND), | (Bitwise OR)

# Where Clause

## Frequently used Operators

AND (to check for both conditions to be true)

OR (to check for one of the conditions to be true)

BETWEEN (selects for a given range)

IN (matches any value in the list)

NOT (to negate the given condition)

# Limit Clause

**Sets an upper limit on number of (tuples) rows to be returned**

```
SELECT col1, col2 FROM table_name
LIMIT number;
```

# Order by Clause

**To sort in ascending (ASC) or descending order (DESC)**

```
SELECT col1, col2 FROM table_name
ORDER BY col_name(s) ASC;
```

# Aggregate Functions

Aggregate functions perform a calculation on a set of values, and return a single value.

- COUNT( )
- MAX( )
- MIN( )
- SUM( )
- AVG( )

Example :

```
SELECT max(marks)
FROM student;
```

# Group by Clause

Groups rows that have the same values into summary rows.
It collects data from multiple records and groups the result by one or more column.

```
SELECT col1, col2
FROM table_name
GROUP BY col_name(s);
```

*Generally we use group by with some *aggregation function*.

# Having Clause

Similar to Where i.e. applies some condition on rows.
But it is used when we want to apply any condition after grouping.

```
SELECT col1, col2
FROM table_name
GROUP BY col_name(s)
HAVING condition;
```

- WHERE is for the table, HAVING is for a group
- Grouping is necessary for HAVING

# General Order

```
SELECT column(s)
FROM table_name
WHERE condition
GROUP BY column(s)
HAVING condition
ORDER BY column(s) ASC;
```

# Table Queries

**Update (to update existing rows)**

```
UPDATE table_name
SET col1 = val1, col2 = val2
WHERE condition;
```

# Table Queries

**Delete (to delete existing rows)**

```
DELETE FROM table_name
WHERE condition;
```

# Table Queries

Alter (to change the schema)

ADD Column

```
ALTER TABLE table_name
ADD COLUMN column_name datatype constraint;
```

DROP Column

```
ALTER TABLE table_name
DROP COLUMN column_name;
```

RENAME Table

```
ALTER TABLE table_name
RENAME TO new_table_name;
```

# Table Queries

Truncate (to delete table's data)

```
TRUNCATE TABLE table_name ;
```

# Table Queries

Alter (to change the schema)

CHANGE Column (rename)

```
ALTER TABLE table_name
CHANGE COLUMN old_name new_name new_datatype new_constraint;
```

MODIFY Column (modify datatype/ constraint)

```
ALTER TABLE table_name
MODIFY col_name new_datatype new_constraint;
```

**TABLE SAMPLE:**

```
CREATE TABLE user(
Id int,
Age int,
Name varchar(30) not null,
Email varchar(50) unique,
Followers int default 0,
Following int,
Constraint check (age>=13)
);
Create table post(
Id int primary key,
Content varchar(100),
User_id int,
Foreign key (user_id) references user(id)
);
```

**Entity relationship diagram**