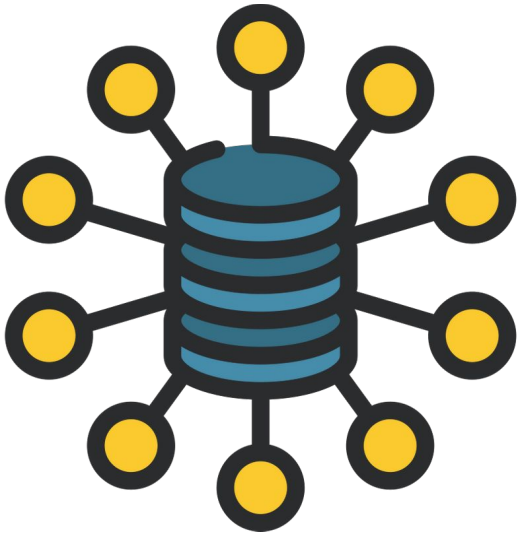
DATABASE SQL

Database, Relational data model
Structured Query Language(SQL)
SQL Commands
Interface of python with SQL

Database

A database is a collection of interrelated data. It's a collection of information that is organised so that it can be easily accessed, managed and updated.

e.g. *A school's database*



Advantages of Database

- Reduces redundancy.
- Controls data inconsistency.
- Facilitates data sharing.
- Ensures data security
- Integrity can be maintained.

DBMS

DBMS(*DataBase Management System*) is a software which helps us establish and implement a database. It consists of a group of programs which helps in storing, retrieving and manipulating data while considering appropriate safety measures.

e.g. *MySQL, Microsoft SQL Server, Oracle, MongoDB, PostgreSQL, etc.*

Relational Data Model

In relational data models, data is organised in tables(*rows and columns*). These tables are called relations.

Name	Roll_No	Marks
Harsh	1	100
Raghav	3	99
Abhinandan	23	01
Bhanupriya	7	00
Hemant	2	73

Terminologies

1. **Relation** : A table storing logically related data.
2. **Domain** : Pool of values from which the actual values appearing in a given column are drawn.
3. **Tuple** : row
4. **Attribute** : Column
5. **Degree** : no. of rows/tuples.
6. **Cardinality** : no. of columns/attributes.

Terminologies

1. **View** : A virtual table that doesn't really exist, but it's used to ***view*** the base table or tables derived from one or more base tables.
2. **Primary key** : One or more attributes that can *uniquely identify* tuples within the table.
3. **Candidate key** : All attribute combinations inside a table that are ***candidates*** to become primary key.
4. **Alternate key** : A candidate key which is not primary key.
5. **Foreign key** : A non-key attribute whose values are *derived* from the primary key of *some other table*, is called foreign key in its *current table*.

SQL(*Structured Query Language*) is a database language that is recognised by nearly all RDBMSs(*Relational DataBase Management System*).
It contains set of commands that enables us to create and operate on relational databases.

Why we use SQL?

SQL is *fast*, *very high level*, *platform independent*, *standardized*, *portable language* which requires almost *no coding skills*.

DDL Commands

DDL or **Data Definition Language**
It contains set of commands that allows us to perform tasks related to data definition.
It specifies the storage structure and access methods for database system.

DDL consists of :

- Creating, altering & dropping.
- Granting, revoking privileges and roles.
- Maintenance commands.

DML Commands

DML or **Data Manipulation Language**
It consists of commands that allows you to perform data manipulation.

DML consists of :

- Insertion
- Deletion
- Modification
- Retrieval

TCL Commands

TCL or **Transaction Control Language**
It consists of commands that allows us to manage and control the transactions (a transaction is one complete unit of work involving many steps)

TCL consists of :

- Making changes to database, permanent.
- Undoing changes to database, permanent.
- Creating savepoints.
- Setting properties for current transactions.

*DCL and DQL are beyond scope of syllabus, we'll study it later sometime.

Important Data Types in MySQL

Data Type	Description
CHAR	string(0-255)
VARCHAR	string(0-255)
TINYINT	integer(-128 to 127)
SMALLINT	integer(-32768 to 32767)
INT	integer(-2^{31} to $2^{31}-1$)
BIGINT	integer(-2^{63} to $2^{63}-1$)
FLOAT	Decimal(precise to 23 digits)

Data Type	Description
DOUBLE	Decimal(24 to 53 digits)
DECIMAL	'DOUBLE' stored as string
DATE	YYYY-MM-DD
DATETIME	YYYY-MM-DD HH:MM:SS
TIMESTAMP	YYYYMMDDHHMMSS
TIME	HH:MM:SS

Varchar

The difference between a char and varchar is of *fixed length* and *variable length*.

When a column is given datatype char(n), then values under this attribute are strings of ***n*** bytes. If a value is less than *n bytes*, then empty spaces are added till *nth byte*.
In varchar(n), values are still strings of ***n*** bytes. No spaces are added even if size is smaller than *n bytes*.

‘Hey’	*char(5)*	*varchar(5)*
	‘Hey ’	‘Hey’

Varchar	Varchar2
• It can identify NULL and empty string separately. • It’s an internal data type. • Minimum size - 1 Maximum size - 2000	• NULL and empty string are same. • It’s an external data type. • Minimum size - 1 Maximum size - 4000

Creating table

CREATE TABLE <table-name> (<column-name> <data type>, .. , .. , ..)

```
CREATE TABLE student
(       name      char(20),
        rollno    int,
        marks     int  );
```

Name	rollno	marks

Inserting data into table

INSERT INTO <tablename> (<column list>)
VALUES (<value>, <value2>, ...);

INSERT INTO student
VALUES ('Harsh', 141, 97);

INSERT INTO student(Name, marks)
VALUES ('Raghav', 99);

INSERT INTO student
VALUES ('Hemant', 146, NULL);

Name	rollno	marks
Harsh	141	97
Raghav	*null*	99
Hemant	146	*null*

Using DATE

CREATE TABLE <table-name>(<column-name> date, .. , ...)

CREATE TABLE transac(Amount int, Date date)

INSERT INTO transac
VALUES (10000, '2021-17-01');

INSERT INTO transac(Amount,Date)
VALUES (-3000, '2021-19-01');

INSERT INTO transac
VALUES (2, '2021-24-01');

Amount	Date
10000	2021-17-01
-3000	2021-19-01
2	2021-24-01

Making Queries

```
CREATE TABLE student
( Name char(20) NOT NULL,
  Rollno int,
  Marks int );
```

Name	rollno	marks
Harsh	141	97
Raghav	*null*	99
Hemant	146	*null*

The **SELECT** command is used to pull information from a table.

```
SELECT what_to_select
FROM which_table
WHERE conditions_to_satisfy;
```

Name
Harsh
Raghav
Hemant

```
SELECT Name
FROM student
```

Making Queries

Name	rollno	marks
Harsh	141	97
Raghav	*null*	99
Hemant	146	*null*

SELECT Name, rollno
FROM student

Name	rollno
Harsh	141
Raghav	*null*
Hemant	146

SELECT Name
FROM student
where marks>95

Name
Harsh
Raghav

SELECT *
FROM student

Name	rollno	marks
Harsh	141	97
Raghav	*null*	99
Hemant	146	*null*

ALL, DISTINCT, BETWEEN

Name	rollno	marks
Harsh	141	97
Raghav	143	99
Hemant	146	97

SELECT ALL Name
FROM student

Name
Harsh
Raghav
Hemant

SELECT DISTINCT marks
FROM student

marks
97
99

SELECT *
FROM student
WHERE marks
BETWEEN 90 AND 98

Name	rollno	marks
Harsh	141	97
Hemant	146	97

90 & 98 are inclusive

Null, ORDER BY

Name	rollno	marks
Harsh	141	97
Raghav	*null*	99
Hemant	146	*null*

```
SELECT Name, rollno
FROM student
WHERE marks IS null
```

Name	rollno
Hemant	146

```
SELECT Name, rollno
FROM student
WHERE marks IS not null
```

Name	rollno
Harsh	141
Raghav	*null*

```
SELECT *
FROM student
ORDER BY marks ASC
```

Name	rollno	marks
Hemant	146	*null*
Harsh	141	97
Raghav	*null*	99

GROUP BY, count, DESCRIBE, DISTINCT

Name	rollno	marks
Harsh	141	97
Raghav	143	99
Hemant	146	97

```
SELECT marks, count(*)
FROM student
GROUP BY marks
```

marks	count(*)
97	2
99	1

```
SELECT DISTINCT marks
FROM student
```

marks
97
99

```
DESCRIBE student
```

Field	Type	NULL	Key	Default	Extra
Name	char(20)	N	PRI		
rollno	int				
marks	int				

This is just an example table, real output might differ.

IN, HAVING

Name	rollno	marks
Harsh	141	97
Raghav	143	99
Hemant	146	97

```
SELECT rollno, marks
FROM student
WHERE Name IN ('Harsh', 'Suraj', 'Khushi')
```

rollno	marks
141	97

```
SELECT marks, count(marks)
FROM student
GROUP BY marks
HAVING count(marks)>1
```

marks	count(marks)
97	2

LIKE

Name	rollno	marks
Harsh	141	97
Raghav	143	99
Hemant	146	97

```
SELECT name
FROM student
WHERE name
LIKE ‘H%’
```

Name
Harsh
Hemant

```
SELECT name
FROM student
WHERE name
LIKE ‘H_m%’
```

Name
Hemant

```
SELECT name
FROM student
WHERE name LIKE ‘%t’
OR name LIKE ‘%gh%
```

Name
Raghav
Hemant

```
SELECT name
FROM student
WHERE name
LIKE ‘%a_s%’
```

Name
Harsh

SQL functions

Name	rollno	marks
Harsh	141	97
Raghav	143	99
Hemant	146	97

SELECT SUM(marks)
FROM student

sum(marks)
293

SELECT AVG(marks)
FROM student

avg(marks)
97.666666….

SELECT productid, max(quantity), min(quantity)
FROM Orders
GROUP BY productid having avg(quantity)>100

SQL Joins

An SQL Join is a query that fetches data from two or more tables whose records are joined with one another based on a condition.

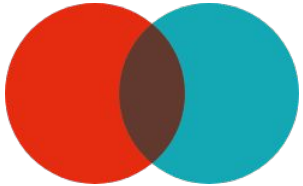
```
SELECT <attributes>
FROM <table1>,<table2>,<table3>, . . . .
WHERE <join condition of tables>
```

e.g.

```
SELECT Name, marks, AdmNo
FROM student, list
WHERE student.name=list.name
```

This will join tables **student** and **list** on the basis of condition student.name=list.name

Types of Joins



1. Cartesian Product/Cross Join :
   Without any condition, returns all joined records.
2. Equi Join :
   Joins two or more tables based on a condition using equality operator.
3. Natural Join :
   A type of equi join where the join condition compares all the same names columns in both tables.

Primary key

A primary key is used to uniquely identify each row in the table. It can be either one attribute or an artificial field(like serial number).

```
CREATE TABLE student
( Name char(20)
  Rollno int PRIMARY KEY
  Marks int );

INSERT INTO student
VALUES ('Bhanu', 141, 33)
```

Name	rollno	marks
Harsh	141	97
Raghav	143	99
Hemant	146	97

Composite key

A **composite key** or **primary key composition** is a combination of two or more columns in a table that can be used to uniquely identify each row in the table.

```
CREATE TABLE student
( Name char(20)
  Class int
  Marks int
  PRIMARY KEY(Class, Marks );
```

Name	Class	RollNo
Harsh	9	80
Raghav	10	80
Hemant	10	87

FOREIGN key

```
CREATE TABLE subject marks
(  rollno int,
   Maths int,
   Science int,
   Hindi int,
   English int,
   FOREIGN KEY (rollno)
   REFERENCES student(rollno)   );
```

Name	**rollno**	marks	class
Harsh	141	97	
Raghav	143	99	
Hemant	146	97	

rollno	Maths	Science	Hindi	English
141	100	100	100	100
143	99	93	97	85
146	80	87	81	33

ALTER TABLE <tablename>

```
CREATE TABLE student
( Name char(20)
  Rollno int
  Marks int );

ALTER TABLE student
ADD PRIMARY KEY (Rollno);
ADD class int
DROP COLUMN Name
MODIFY COLUMN class char(5)
```

Name	**rollno**	marks	class
Harsh	141	97	
Raghav	143	99	
Hemant	146	97	

THANK YOU FOR WATCHING!

Milte hain next video me, BYEE!!!!