SQL

----

1. What is database ?

   :- Database is a collection of a relational database.

2. What is relational database?

   :- collection of relational databases.

3. DBMS longform?

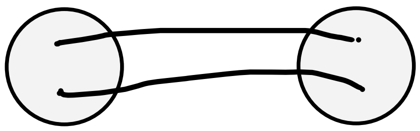
   :- Database management system. (DBMS nothing but the software that manages the data.)

4. What is attribute ?

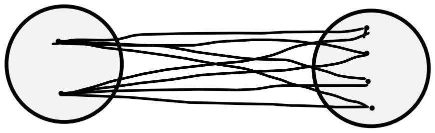
   :- Attribute is a characteristics or property of an given entity.

5. Types of relationship ?

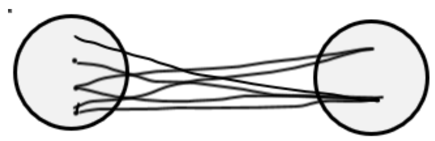
   - 1 to 1



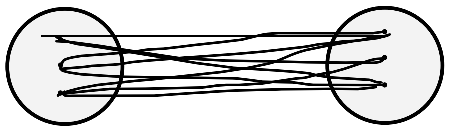
   - 1 to many



- Many to 1



- Many to many



: Keys :

It is an attribute which identifies the record or tuple uniquely.

Name	Aadhar	Pan	Passport	License	address	State
Pratik	0725	165		9462		

Combinations of attributes

Name + Aadhar	Name + Pan +License

- Candidate key

  Combination/set of 1 or more attributes which is used to identify record/tuple uniquely.

- Primary key

  It is a key selected for unique identification of tuple, which cannot have duplicate value or null value.(One of between the candidate or eligible keys for the unique identification).

- Alternate key

  Remaining candidate keys other than a primary key.
- Composite/compound key

  Set of more than 1 attributes which identifies tuple uniqueness.
- Foreign key

  So, Foreign key is used to establish relationship between 2 tables and usually it is a primary key of another table used to establish association between entities (Tables) relationship.
- Super key

  A super key is a key supreme (Super-set)of a candidate keys.
- Parent key

  It is either primary key unique key in a parent table or referential constraint.

6. What is normalization ? *IMP Question :*

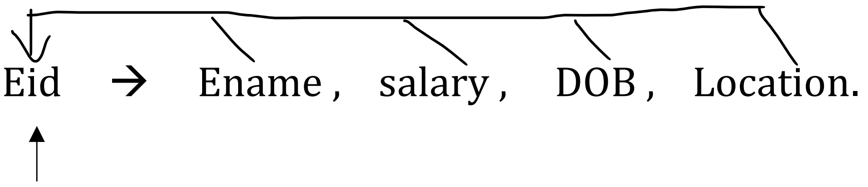
   It is a database design technique that reduces data redundancy and ensures that data is stored logically.

   It divides longer tables into smaller tables logically and links them by establishing association among them.

: Functional dependency :

Emp

Eid	Ename	Salary	DOB	Location


Eid → Ename , salary , DOB , Location.

Uniquely identifies emp record.

Ex : B → A : A is functionally dependent on B.

B = Determinant

A = Dependent.

Here , 'A' is functionally Dependent on 'B'.

Here , 'B' Determines (Identifies) 'A'.

Roll no	C -code	Cname	Trainer	Room	Marks	Grade
1	101	Java	Vinayak	R001	75	A
2	102	SQL	Pallavi	R002	95	A
3	102	SQL	Pallavi	R002	85	A
4	103	Apti	Girija	R003	65	B
5	102	SQL	Pallavi	R002	90	A
A	B	C	D	E	F	G

B → C ,D	A,B → F
D→ E	
F → G	

Primary key ← A,B → C (C is partially dependent on AB)

- Partial dependency :

  AB → C (But C is only dependent on B)

  There is no need of A to determine C.

  :- So, when an attribute Is not completely dependent on a complete key, it is dependent or part of a key.

- Fully-Functional Dependency :

  AB → F (F is fully dependent on AB)

  Only A or Only B cannot determine F.

  :- So, when an attribute depends on each part of primary key for unique identification.

- Transitive dependency :

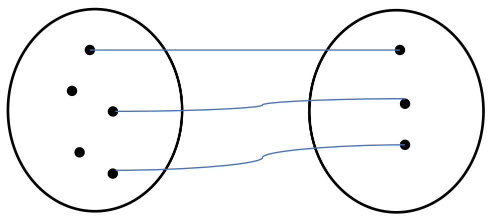
Couse Code		Trainer name		Room No	
101	Java	Vinayak	Java	R001	Vinayak
102	Apti	Pallavi	SQL	R002	Pallavi
103	Apti	Girija	Apti	R003	Girija

$$\begin{tikzcd}\text{Course Code} \arrow[r] & \text{Trainer name} \arrow[r] \arrow[l, bend left] & \text{Room No} \arrow[l, bend left]\end{tikzcd}$$

Transitive Dependency

When a determinant depends on another determinant called transitive dependency.

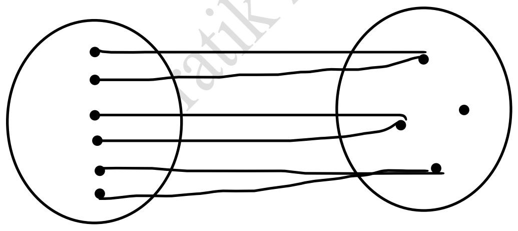
1} 1 : 1 : Emp Passport $\longrightarrow$ Having Total Participation



Emp(Eid, name, salary) Primary key(Emp)

Passport(pid, passport_number, type, Eid)

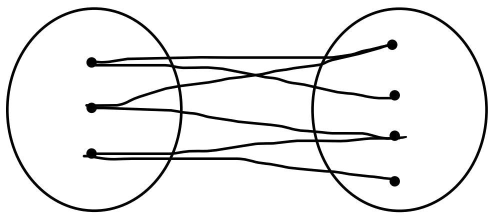
2} 1:M | M:1 Emp Dept



Emp(Eid, ename, sal, Did)

Dept(Did, name, count) $\uparrow$ Primary key(Dept)

3} M:M : Student Course



Student(RollNo, name, ~~contact~~, cid)

Course(cid, cname, ~~fees~~, RollNo)

Primary key (Student, Course)

$$\text{Stud\_Course(scid, }\overset{\downarrow}{\text{Roll}}\text{, }\overset{\downarrow}{\text{cid}}\text{, PaymentDate)}$$

: Normal Forms :

- 1NF :

  No Data redundancy

- 2NF :

  No partial dependency.

  Must fit in 1NF.

- 3NF :

  Remove transitive dependency.

  Must fit in 2NF.

- BCNF :

  Must fit in 3NF.

  For each functional dependency, say X → Y; X must be the subset of super key.

  i.e., LHS must be the candidate key.

---X--------------------------------------

: Commands :

DDL : (Data definition language) : (Database designing)

- Create Database
- Delete Database
- Table Create (Table name, Fields, It's Datatype)

- Table Delete
- Table Update (Field name / Datatype update)

DML : (Data manipulation language) :

- Insert data
- Update data
- Delete data

DQL : (Data Query Language) :

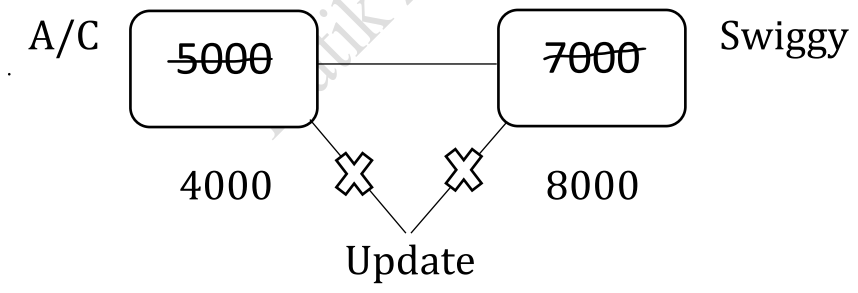
- Fetching data from tables.

DCL : (Data Control Language) :

- Controls the privileges of database.

TCL : (Transition Control Language) :

- Either while transaction should be executed or none of it.



Create database :

Create database database_name;

Show database :

Show databases;

Use database :

Use database_name;

Display tables :

Show tables;

Describe Table :

Desc table_name;

Create table :

Create table table_name

(<Field_name 1> <datatype>

<Field name 2><datatype>);

Modify table: (change field name or datatype in structure)

Alter table table_name

Modify <Field_name><datatype>;

Add column table :

Alter table table_name

Add column <column_name> <datatype>,

Add column <column_name> <datatype>;

Delete column :

Alter table table_name

Drop column_name,

Drop column_name;

Add column after some other column :

```
Alter table table_name

Add column <column_name> <datatype>

After <column_name>;
```

Add column at 1st position :

```
Alter table table_name

Add column <column_name> <datatype>

first;
```

Rename table name :

```
Alter table table_name

Rename to <new_table_name>;
```

Change column name :

```
Alter table table_name

Change <column_name><new_column_name><datatype>;
```

Delete table :

```
Drop table table_name;
```

Delete data inside table (Not table) :

```
Truncate table <table_name>;
```

Delete database :

```
Drop database database_name;
```

Insert value into table (After creation of table):

```
Insert into table_name(field1,field2,field3)
values(value1,value2,value3);
```

Change values at specific location : (Under single condition)

```
Update table_name set column_name = value
Where column_name = value;
```

Change values at specific location : (Under double condition using and)

```
Update table_name set column_name = value
Where column_name = value and
2nd_column_name = value;
```

Change values at specific location : (Under single condition using or)

```
Update table_name set column_name = value
Where column_name = value or
```

2nd_column_name = value;

Interview Question of any 2 comparison :

Drop - DDL :

- Deletes table, database, and field along with data
- Autocommit , cannot restore with rollback.

Delete - DML :-

- Deletes data.
- ‘where’ clause can be used.
- Slower than truncate.
- Can be restored with rollback.

Truncate – DQL :-

- Deletes all the data.
- Cannot use ‘where’ clause.
- Faster than delete.
- Autocommit, hence cannot be restored with rollback.

Create copy of any another table :

Create table table_name as select * from another_table_name;

Give alice (nickname) to the table_name : table_name alice;

: Constraints :

It is rule enforced on data being added into the tables.

1. UNIQUE :

    When table is already created :

        Alter table table_name

        Modify <column_name><datatype> UNIQUE;

    Table creation :

        Create table table_name

        (<Field_name 1> <datatype> UNIQUE

        <Field name 2><datatype>);

2. NOT NULL :

    When table is already created :

        Alter table table_name

        Modify <column_name><datatype> NOT NULL;

    Table creation :

        Create table table_name

        (<Field_name 1> <datatype> NOT NULL

        <Field name 2><datatype>);

3. PRIMARY KEY :

    When table is already created :

Alter table table_name

Modify <column_name><datatype> PRIMARY KEY;

Table creation :

Create table table_name

(<Field_name 1> <datatype> PRIMARY KEY

<Field name 2><datatype>);

4. DEFAULT :

When table is already created :

Alter table table_name

Modify <column_name><datatype>

DEFAULT(VALUE);

Table creation :

Create table table_name

(<Field_name 1> <datatype> DEFAULT(VALUE)

<Field name 2><datatype>);

5. CHECK :

When table is already created :

Alter table table_name

Modify <column_name><datatype>

CHECK(CONDITION);

Table creation :

Create table table_name

(<Field_name 1> <datatype> CHECK(CONDITION(< , > , - , +))

<Field name 2><datatype>);

Drop constraint : (remove constraint)

Alter table table_name

Drop constraint constraint_name;

Add multiple condition in check or anywhere : (use IN)

Alter table table_name

Add check(column_name IN (value1, value2, ...));

6. FOREIGN KEY :

One table primary key can be foreign key for multiple tables.

Alter table table_name

Add column <column_name><datatype>,

Add foreign key (column_name) references

another_table_name(referred_column_name)

$\uparrow$ *Primary key of another table*

Change/set CONSTRAINT name :

Alter table table_name

Add CONSTRAINT constraint_name CHECK(cond'n);

Show column as another name(only show) :

Select column_name as temp_column_name from table_name;

Work if we don't use (as) but it's not a good practice ☹.

Show data using between : (middle data of a range)

Select * from table_name where column_name BETWEEN value1 and value2;

Sorting data using order by : (ascending or descending)

Select * from table_name order by column_name;

By default it has ascending order (or use ASC).

Select * form table_name order by column_name DESC;

Multiple condition for sort –

Select * from table_name order by column_name, another_column_name;

Get distinct value form table :

Select DISTINCT (column_name) from table_name;

Get null values from table :

Select * from table_name where column_name IS NULL;

Get not null values :

Select * from table_name where column_name IS NOT NULL;

Favorite interview questions :

- *SHOW THE DATA LIKE HIGHEST SALARY OF A EMPLOYEE*

Select limited data from table :

Select * form table_name LIMIT <VALUE>;

Skip record for next record

- *SHOW THE DATA LIKE 2ND HIGHEST SALARY OF A EMPLOYEE*

Select * from table_name LIMIT<VALUE,SKIP_ROWS_NO>;

: Group Functions : (aggregate functions) :

Sort data by (MIN,MAX,SUM,AVG) :

MIN : Select MIN(column_name) from table_name;

MAX : Select MAX(column_name) from table_name;

SUM : Select SUM(column_name) from table_name;

AVG : Select AVG(column_name) from table_name;

Get count of all records present in table :

Select COUNT(*) from table_name;

OR

Select COUNT(column_name) from table_name;(remove null's)

Select data from table using GROUP BY :

Select column_name from table_name group by column_name;

Having clause in group by :

Select column_name from table_name group by column_name

Having column_name <Condition><value>;

Multiple select from table :

Select * from table_name where column_name <| =/>/< |>

(select column_name from table_name where column_name=value);

For multiple values use (IN) instead of (=).

Get multiple values for : (< / >)

When you want to compare multiple values in (< less than or > Greater than) use (ANY) keyword.

Select * from table_name where column_name (< / >) ANY

(select column_name from table_name where column_name = value);

-: Types of joins :-

- Cartesian Join/Cross Join.
- Natural / Equi / Inner Join.
- Outer Join –
    - Left Outer Join.
    - Right Outer Join.
    - Full Outer Join.
- Self Join.

1} Cartesian Join :

TABLE A

Col A1	Col A2
A1	A11
A2	A22
A3	A33

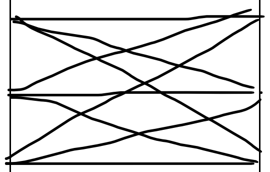


TABLE B

Col B1	Col B2
B1	B11
B2	B22
B3	B33

Get all records from both the tables

: OUTPUT :

Col A1	Col A2	Col B1	Col B2
A1	A11	B1	B11
A1	A11	B1	B11
A1	A11	B1	B11
A2	A22	B2	B22
A2	A22	B2	B22
A2	A22	B2	B22
A3	A33	B3	B33
A3	A33	B3	B33
A3	A33	B3	B33

2} Inner Join :

TABLE A fk pk TABLE B

Col A	Col B
A1	B1
A2	B2
A3	B3

Col B1	Col C
B1	C1
B1	C2
B2	C3
B2	C4
NULL	C5

<u>Get all the matching records from both the tables</u>

: OUTPUT :

Col A	Col B	Col B	Col C
A1	B1	B1	C1
A1	B1	B1	C2
A2	B2	B2	C3
A2	B2	B2	C4

3} Outer Join : Left Outer Join

TABLE A fk pk TABLE B

Col A	Col B
A1	B1
A2	B2
A3	B3
A4	NULL

Col B1	Col C
B1	C1
B1	C2
B2	C3
B2	C4
NULL	C5

Get whole left table records and get the matching records from the right table

: OUTPUT :

Col A	Col B	Col B	Col C
A1	B1	B1	C1
A1	B1	B1	C2
A2	B2	B2	C3
A2	B2	B2	C4
A3	B3	NULL	NULL
A4	NULL	NULL	NULL

3} Outer Join : Right Outer Join

TABLE A fk pk TABLE B

Col A	Col B
A1	B1
A2	B2
A3	B3
A4	NULL

Col B1	Col C
B1	C1
B1	C2
B2	C3
B2	C4
NULL	C5

Get whole right table records and get the matching records from the left table

: OUTPUT :

Col A	Col B	Col B	Col C
A1	B1	B1	C1
A1	B1	B1	C2
A2	B2	B2	C3
A2	B2	B2	C4
NULL	NULL	NULL	C5

3} Outer Join : Full Outer Join - Left + Right Outer Join

Non - matching records from both tables + matching records from both tables.

: Union :

```
(select * from left_table_name left outer join right_table_name
on right_table_column_name = left_table_column_name)
union
(select *from left_table_name Right outer join right_table_name
on left_table_column_name = right_table_column_name );
```

It is a keyword used to combine the output of 2 Queries.

4} Self Join :

EMP

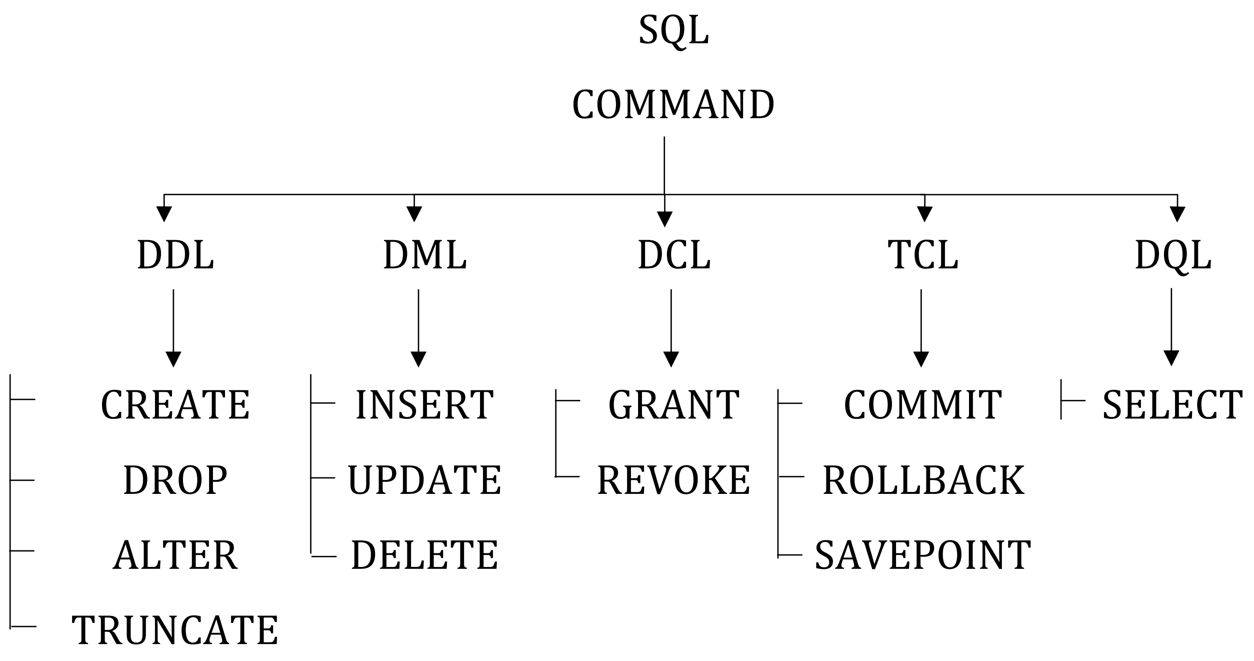
1st 2nd

Eid	Ename	Mgr-id
1	Pranali	5
2	Kaveri	4
3	Vishal	1
4	Anokh	3
5	Shubham	2
6	Harshal	1

select <table_alice_name> . <column_name> <column_alice_name> , <table_alice_name> . <column_name> <column_2nd alice_name>

from <table_name><1alice name> , <table_name><2alice name>

where <1alice_name> . <2column_name> = <2alice_name> . <1column_name>;


SQL
COMMAND
DDL
DML
DCL
TCL
DQL
CREATE
DROP
ALTER
TRUNCATE
INSERT
UPDATE
DELETE
GRANT
REVOKE
COMMIT
ROLLBACK
SAVEPOINT
SELECT