2. Structured Query Language (SQL)

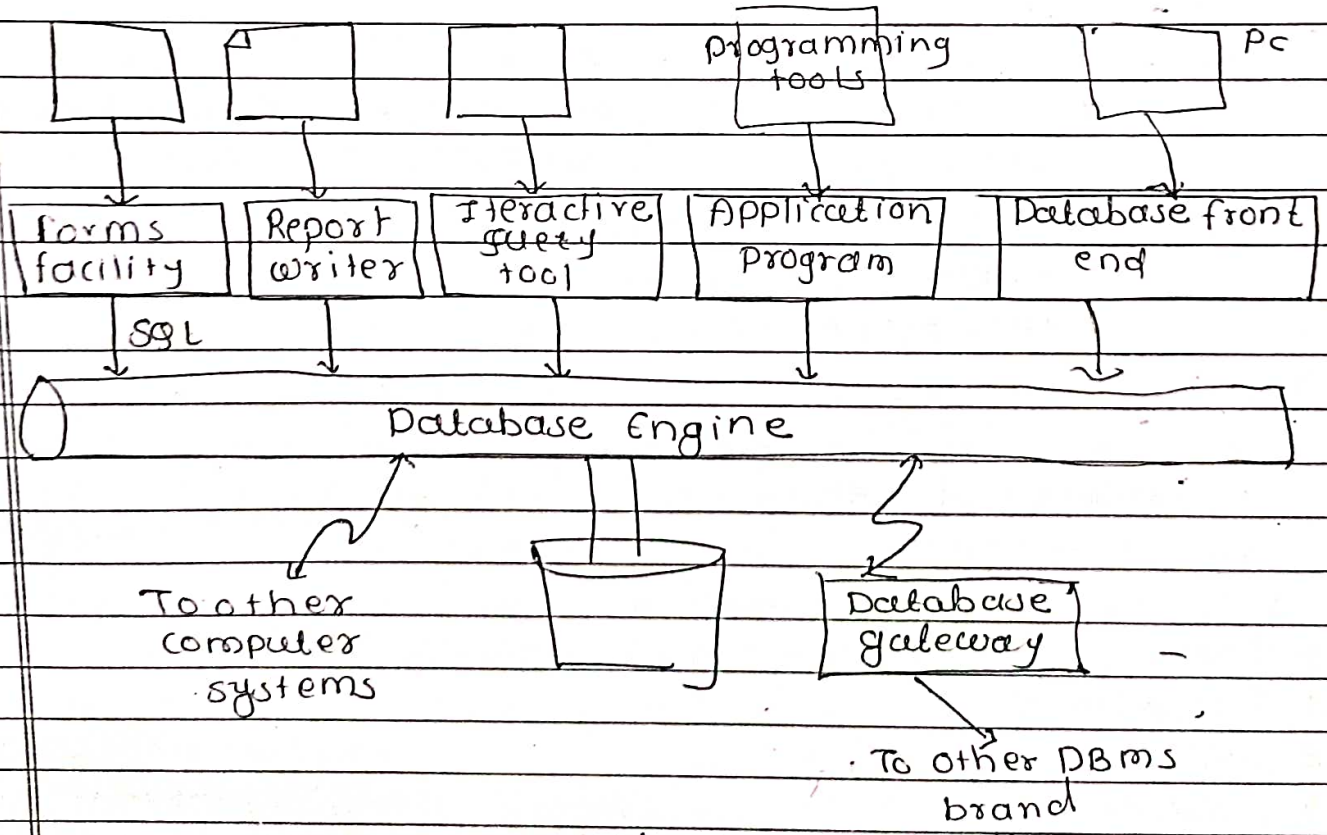
Page No.
Date / /

Introduction (SQL)

It is a computer language used to store manipulate & retrieve data stored in relational databases.

It is a keyword based language & each statement begins with a unique keyword

- SQL syntax is not case sensitive
- SQL in an interactive query language which can be used to retrieve data from database, & used to access data from database
- SQL is a database administration language which can be used to monitor & control data access by various users.
- SQL can be used as an Internet data access lang



Components of typical DBMS

SQL Schema

The structure of database which is specified at time of database design is called as database schema

e.g → Student table

Sid	Name	class

Course table

Cid	Name	hours

Department table

Did	Name

Marks table

Sid	Cid	Marks	Grade

Student Database

Features

- Database schema is not expected to change frequently
- Database schema can be represented as a schema diagram
- Each object in the schema - such as Student or course is called as schema construct.
- A schema diagram displays only some parts of a schema such as the names of data items & some types of constraints on data items.
- many types of constraints are not represented in schema diagram.

Student database from above students schema

Student table

Sid	Name	class
1	Sachin	FE
2	Jayendra	TE
3	Mahesh	SE
4	Suhas	SE

course table

~~Code~~ Cid	Name	Hours
10	DBMS	120
20	ADBMS	150
30	DW	120
40	DM	80

Page No. / Date / /

Department table

Did	Name
10	IT
20	CSE

Marks Table

sid	cid	marks	Grade
1	10	90	A
2	20	97	A
3	30	99	A
4	40	89	B

fig - schema diagram for the Database

1.4 * Data Definition Language (DDL) — To specify schema

- To create database schema we use Data definition language (DDL).
- DDL statements are used to build & modify the structure of your tables & other objects in the database.

It is a set of command that are used to

1) Create
2) Alter
3) Drop
4) Rename
5) Truncate.

(DRACT)

when you execute DDL statement it takes effect immediately as it is autocommitted into database. Hence no rollback operation (Undo) can be performed with these set of commands.

- Database objects are any data structure created in database.

e.g → Table, view, sequence etc

① CREATE command

· This statement used to create database obje

Syntax –

```
CREATE TABLE <Table_Name>
 (Column_1 datatype,
  column_2 datatype,
    :
  column_n datatype
 );
```

Example

```
CREATE TABLE Employee
  (Eid varchar2(10)
   First_Name char(50)
   Last_Name char(50);
   Address char(50)
   City char(50)
   Birth_Date date
  );
```

To view the structure of table

DESCRIBE TABLE Employee

② DROP command

- This command can be used to remove database objects from our DBms.

· DROP command removes data from your database

Syntax → DROP TABLE <Table_Name>

Example →

If we want to permanently remove the Employee table that we created, we'd use the following command

DROP TABLE Employee

A DROP statement cannot be rolled back, so once an object is destroyed, there's no way to recover it.

- similarly the command below would be used to remove the entire employees database

```
DROP DATABASE Employees
```

(3) ALTER command

- Once database object is created in database sometimes, we may require to change structure of database object. This can be done with the help of ALTER command.
- The alter table statement may be used as you seen to specify primary & foreign key constraints, as well as to make other modifications to the table structure

Syntax

```
ALTER TABLE <Table-Name>
ADD column_1 datatype;
```

OR

```
ALTER TABLE <Table-Name>
modify column_1 New-datatype;
```

OR

```
ALTER TABLE <Table-Name>
DROP column_1;
```

Example →

```
ALTER TABLE Employee
ADD Address Varchar2(100);
```

To view the changed structure of table

```
DESCRIBE TABLE Employee
```

(4) RENAME command

We can rename any database object at any point time.

Syntax

```
RENAME TABLE <Table-Name>
To <New-Table_Name>;
```

Example → RENAME TABLE Employee
To EMP;

(5) Truncate Table

- Use the truncate statement to delete all the rows from table permanently.
- It works same as 'DELETE FROM <table-name>' but Truncate can not be rolled back (undo)
- If any delete triggers are defined on the table, then the triggers are not fired on truncate table
- It deallocates memory space, so that the free space can be used by other tables.

Example

```
TRUNCATE TABLE EMP;
```

If you do not want to free space & keep it with the table, then specify the REUSE storage clause like following query.

```
TRUNCATE TABLE EMP REUSE STORAGE
```

Page No.
Date / /

* Data Manipulation Language (DML) → To express DB queries & update

It is set of commands used to

i- Insert data into table
ii - Delete data from table
iii- Update data of table

A language that provides a set of operations to support the basic data manipulation operations on the data held in the database

i) INSERT

Insert statement used to add records to the existing table.

syntax INSERT INTO <table_Name> [(column1 ..... N)]
VALUES (column1 ....., columnN);

example INSERT INTO Employee
VALUES (100, 'swati');

INSERT TNTO Employee
VALUES (NULL, Jay);

INSERT INTO Employee (Name, Eid)
VALUES ('Swarga', 1001)

INSERT INTO Employee (Name, Eid)
VALUES ('Sarei', Null);

To check inserted rows write following query

SELECT * FROM Employee

Eid	Name
100	Swabga
1001	Swara
NULL	Jay
NULL	...

This command is used to enter the information or values into a row. We can connect one or more records to a single table within a repository using this instruction. This is often used to connect an unused tag to the documents.

(3) DELETE

Delete statement is used to delete some or all records from the existing table

syntax

```
DELETE
FROM <Table-Name>
WHERE condition;
```

example →

```
DELETE
FROM employee
WHERE Eid is NULL;
```

To check inserted rows write following query

```
SELECT *
FROM Employee
```

Eid	Name
100	swati
1001	sweta

(4) Update

Insert statement used to modify the record present in existing table

syntax

```
UPDATE <Table-Name>
SET    column1 = value
WHERE condition
```

Example —

```
UPDATE employee
SET Eid = 1008
WHERE name = 'swetha';
```

Page No.
Date / /

To check inserted rows write following query

```
SELECT * FROM employee;
```

Eid	Name
100	swati
1001	swaed
1003	swapnali
NULL	sw Jay

* Data control Language (DCL)

- To allow or restrict users from accessing data schema we use data control language (DCL). It used to grant or revoke access permissions from a database user or database role.
- In real time database applications having multi users with same access rights, need to have same privileges. Eg - there are many users who are of type managers & having same access rights on system in company.
- Instead of giving grant to each & every manager users we can create a generalized set of grant privilees with name manager & this can be assigned to all employees belongs to category manager such concept is called as Role.
- A set of role can be created in a DB & assign to any new users entered in database system as per authorization required.
- It is a set of command that are used to

a) GRANT access to any database objects

b) REVOKE access rights given to any database objects.

(a) GRANT statement

A system privilege is the right to perform a particular action, or to perform an action on any schema objects of particular type.

. An authorized user may pass on this authorization to other users. This process is called as granting of privileges.

Page No.
Date / /

syntax

REVOKE <privilege list>
ON <relation name or view name>
From <user / role list> [District/cascade

Example → The revocation of privileges from user or role may cause other user or roles also have to loose that privilege. This behaviour is called cascading of the revoke.

Revoke select
ON Emp-Salary
FROM U_1, U_2, U_3

Revoke Update (amount)
ON Emp-Salary
FROM U_1, U_2, U_3

Revoke references (amount)
ON Emp-Salary
FROM U_1

REVOKE select
ON Emp-Salary
FROM U_1, U_2, U_3 restrict

The following revoke statement revokes only the grant option rather than the actual select privile

```
REVOKE grant option select
ON Emp-Salary
FROM U1
```

* Transaction control Language (TCL)

Any SQL query can be executed with the two basic operations on the database objects

a) Read
b) Write

- After executing SQL query we must specify it's final action as commit (save data) or abort (or revert back changes)
- The COMMIT statement ends the operations & makes all changes made to the data permanent, on successful completion. ABORT terminates & undoes all the actions done so far.

i) Commit transaction

A query that is successful & has encountered no errors is committed by issuing commit. That is all changes to the database are made permanent & become visible to other users of the database

syntax

```
COMMIT (WORK)
```

The keyword WORK is not required, though it might be added for clarity. a simple COMMIT is usually all that is required.

```
COMMIT [TRAN [SACTION] [transaction name]]
```

Page No.
Date / /

Syntax

GRANT <privilege list>
ON <relation name or view name>
TO <user/role list>

Example → consider an example for granting update authorization to the Emp-Salary relation of the company DB. Assume that initially that the DBA grants update authorization on Emp-Salary to other users U_1, U_2 & U_3 who may in turn pass on this authorization to other users.

- This passing of authorization from one user to other users is called authorization graph.

- The following grant statement grants user U_1, U_2 & U_3 the select privilege on Emp-Salary relation

```
GRANT Select
ON Emp-Salary
To, U1, U2 & U3
```

- Following grant statement gives all users all authorization on the amount attributes of the Emp-Salary relation using public keyword

```
GRANT ALL
ON Emp-Salary
To PUBLIC
```

- To view privileges given to table

```
SELECT GRANTEE, OWNER, GRANTOR, PRIVILEGE
    GRANTABLE
FROM DBA_TAB_PRIVS
WHERE TABLE_NAME = 'Emp-Salary';
```

Types of privileges

i) Reference privileges

- SQL permits a user to declare foreign key while creating relations.

Example → Allow user U_1 to create relation that reference key 'Eid' of Emp-Salary relation.

```
GRANT REFERENCES (Eid)
    ON Emp-Salary
    TO U1
```

ii) Execute privileges

This privilege authorize a user to execute function or procedure.

Thus, only user who has execute privilege on a function Create_Acc() can call function.

```
GRANT EXECUTE
ON Create_Acc
TO U1
```

(b) REVOKE

We can reject the privileges given to particular user with the help of revoke statement.

To revoke an authorization, we use the revoke statement

- COMMIT statement is required field others is optional & the keywords can be shortend i.e TRAN instead of TRANSACTION. Alternatively COMMIT WORK can be used.

ii) Rollback transaction

- A query that is unsuccessful & has encountered some type of error should be rolled back. That is all changes to the database are undone & the database remains unchanged by the transaction.
- Transaction processing systems ensure database integrity by recording intermediate states of the database as it is modified, then using these records to restore the database to a known state if a transaction cannot be committed.

Syntax ROLLBACK

iii) Savepoint

- A query that is in process of execution & these partial results needs to be commited to avoide problem of transaction failure before final commit.
- A savepoint will save all data executed before it.
- Savepoint can be used for partial commit, so all changes before savepoint are commited & changes after savepoint are rolled back.

Syntax SAVEPOINT <savepoint_name>

* Data Retrieval Language (DRL) / Data selection Language (DSL)

- DRL is a set of commands used to retrieve data from database server.
- This language also includes all functions used to manipulate data in database for display purpose like aggregate functions.

⊙ Basic SQL queries

i) **select clause** - This clause is used for selecting various attributes or columns of a table.

```
SELECT { * | , column1, column2, ... }
FROM   < table list >
WHERE  < condition >
```

ii) **FROM clause** - This clause is used to select a relation / table name in a database.

iii) **WHERE clause** → This clause is used to ~~select a relation / table name in a database~~ put a condition on query result.

Example

```
SELECT  *
FROM    employee
WHERE   Age > 50
```

In above given query the first executable statement is FROM, this statement will load the all rows in EMPLOYEE table on to server to local memory.

Page No.
Date / /

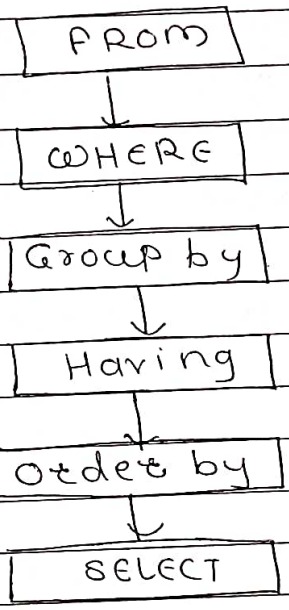
iv) HAVING clause

It acts like a group filter with the help of condition (department_id = 50) then it selects the rows of EMPLOYEE table having dept-id = 50 & other groups are deleted from memory

v) ORDER BY clause

This clause arrange output in proper order than can be either ascending or descending order

Execution hierarchy



* Aliasing in SQL / Naming in SQL

*) Alias → This is method of giving alternate to database object (table) or column or

a) Table alias

- This is accomplished by putting an alias directly after the table name in the FROM clause.
- This is convenient way to obtain information from multiple tables. Table alias is also useful when table name is of longer length to which we can assign a shorter name for simple use. This alias will not perform any operations on table.

syntax :

```
SELECT *
FROM Table-name "Table-alias";
```

Example

```
SELECT Student.Student-code
Department.Department-Name
FROM ~~Faculty~~ Student, Department
WHERE Department.Department-code =
Department.Student-code;
```

Student.Student-code	Department.Department-Name
1	IT
2	Comp
3	E & Tc
4	mech
5	chemical

b) Column alias

- Column aliases exist to help organizing output
- In the previous example, whenever we see ~~Faculty~~ Student-code it is listed as Student-code. Column alias will make the output much more readable by proper column naming.

Page No.
Date / /

syntax

```
SELECT Table-alias.column-name1'
                "column-alias"
FROM Table-name "Table-alias".
```

Example :

```
SELECT S.Student-code "Student code"
       D.Department_Name "Department Name"
FROM ~~Faculty~~ student S, Department D
WHERE S.student-code = D.student-code
```

Student code	Department_Name
1	IT
2	CSE
3	E & Tc
4	mech
5	chemical

ORDER By clause

The ORDER By ~~clau~~ keyword is used to sort the result-set by specified column. It sorts the records in ascending order by default. If you want to sort the records in a descending order, you can use the DESC keyword.

syntax

```
SELECT column-name
FROM table-name
ORDER BY column-nane ASC | DESC
```

Example → student table

S ID	StudntName	DOB
1	Yogesh	17/7/64
2	Amit	24/12/72
3	Om	3/2/80
4	Nitin	28/11/66

then

```
SELECT *
FROM student
ORDER BY student-Name
```

OR

```
SELECT *
FROM student
ORDER BY 2
```

then created table is

S-Id	Student-Name	DOB
1	Amit	24/12/72
2	Nitin	28/11/66
3	Om	3/2/80
4	Yogesh	17/7/64

→

```
SELET *
FROM student
ORDER BY student-Name DESC
```

S-Id	Student-Name	DOB
1	Yogesh	17/7/64
2	Om	3/2/80
3	Nitin	28/11/66
4	Amit	24/12/72

Page No. / Date / /

Aggregate function

-Sometimes for decision making we need summarize data from table like average, sum, minimum etc. SQL provide various aggregate functions which can summarize data of given table. The function operates on the table data produces a single output. Such queries are generally used for producing reports & summary forms in an application

Types

a) COUNT([DISTINCT] C) → The no. of unique values in column C

b) SUM([DISTINCT] C) → The sum of all unique values in the column C

c) AVG([DISTINCT] C) → The average of all unique values in the column C.

d) MIN(C) - The minimum value in the column C

e) MAX(C) - The maximum value in the column C

Example Table → Exam_marks

Sid	SName	Marks
1	A	90
2	B	80
3	C	89
4	D	99
5	E	88
6	F	90

SQL

a) COUNT ()

This function is used to calculate no. of in table selected by query. COUNT returns th of rows in the table when the column value is NULL

Example → i) find total no. of student in Exa marks table

```
SELECT count(Sid) as count
FROM exam-marks
```

→

COUNT
6

ii) Find total no. of different student

```
SELECT count(Distinct Sid) as count
FROM exam-marks;
```

→

COUNT
6

(b) SUM ()

This function is used to calculate sum of column values in a table selected by query. Column in the query must be numeric. Value of the sum must be within the range of that data type.

Example → Find total of marks scored by all student

```
SELECT SUM(marks) as Sum
FROM exam-marks;
```

→

SUM
536

Page No.
Date / /

(c) AVG ()

This function is used to calculate Average of column values in a table selected by query. This function first calculates sum of column & then divide by total number of rows. AVG returns the average of all the values in the specified column. column in the query must be numeric.

Example → Find Average marks of student

```
SELECT AVG (marks) as AVG
FROM Exam-marks;
```

→

AVG
89.3

(d) MIN()

This function is used to find minimum value out of column values in a table selected by query column in the query need not be numeric data type.

Example → Find minimum marks scored by students.

```
SELECT MIN (marks) as min
FROM Exam-marks;
```

→

min
80

(e) MAX()

This function is used to find maximum value out of column values in a table selected by query. Column in the query need not be numeric data type

Example → Find maximum marks scored by

```
SELECT MAX(marks) as max
FROM Exam-marks;
```

MAX
98

* Aggregate functions with group by query

Aggregate functions can be used for some group by clause in this case it will give summary information for that specific group.

Example → Display the maximum salary of all departments in company

```
SELECT department_id as Dept,
       AVG(salary) as Avg
FROM   employees
GROUP By department_id
```

→

DEPT	AVG
10	40000
20	25000
30	35000

Page No. / Date / /

* Referential Integrity

- A value appearing in one relation (table) for a given set of attributes also appears for another set of attributes in another relation (table) This is called referential integrity.
- The referential integrity constraint is specified between two tables & is used to maintain the consistency among tuples in the two relations.
- The tuple in one relation refers only to an existing tuple in another relation.

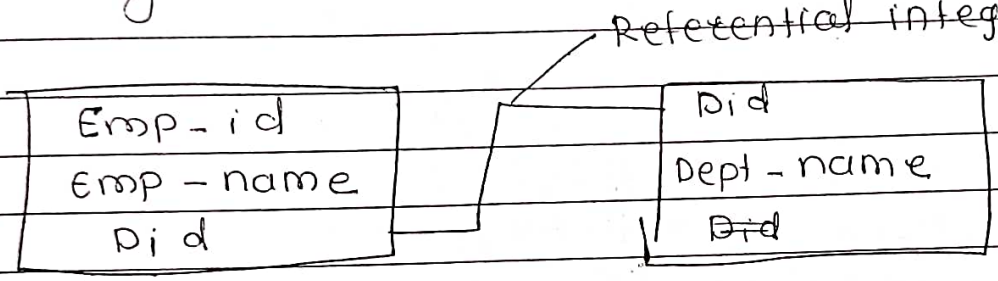


fig → Referential integrity

Emp Table

Emp-Id	Emp-Name	Did
1	A	20
2	B	10
3	C	20
4	D	10

Department table

Did	Dept-name
10	HRE
20	CTIS
80	L&D

In the above example 'Emp' table has Did as foreign key reference this called as referential integrity.

Here we are forcing database to check the 'Did' value key from the 'department' table while inserting any val of 'Emp' table in Did column if there is no value existing department table of that 'Did' then we can not insert th values in 'Emp' table. This helps to maintain data. consi

Referential integrity in SQL

To show the relation into two tables in relational algebra, we use foreign key constraint. This helps us to maintain consistency in database as one can not insert, update or delete foreign key.

syntax -

```
FOREIGN KEY
  REFERENCES [schema_name.] referenced_table_nam
                                [(ref_column)]
  [ON DELETE { NO ACTION
             | CASCADE
             | SET NULL
             | SET DEFAULT } ]
  [ON UPDATE { NO ACTION
             | CASCADE
             | SET NULL
             | SET DEFAULT } ]
```

Example →

```
Create table Emp( Emp_id integer,
        Emp_name varchar(100) not null
        Did as integer,
        Primary key (Emp_id),
        Foreign Key (Did) references department
       on delete cascade
       On update cascade
)
```

Page No.
Date / /

Create table Department (Did integere,
Dept_name varchar (100) not null,
Primary key (did).
)

Self referential relations

A FOREIGN KEY constraint can reference column in same tables or in the same database (self referencing table)
for example an employee table that contains 3 columns employee_number, employee_name & employee_manager. Because the manager is an employee too, there is a foreign key relationship from the employee_manager column to the employee_number column in same table.

DELETE & UPDATE rules

If any row in EMP table is added with 'Did' value which is not there is department table the insert statement will give foreign key violation error.

Foreign Key (Did) references department
On delete cascade
On Update cascade
Any row in department is deleted (ON DELETE)

a) NO ACTION / RESTRICT

In this case the database engine will not allow user to delete the row & restrict the deletion. The restriction rule prevents you from deleting a row from the parent table if there is corresponding row is present in child table.

The database engine raises an error & the action on the row in the parent table is called Deletion of 'Did' is not allowed as in 'Emp' ta that 'Did' may be present.

(b) CASCADE

- corresponding rows are deleted from the refer table if that row is deleted from the parent ta
- Foreign key (Did) references department.
  - On delete CASCADE
  - On Update CASCADE

Delete all the staff record that refers to the deleted dep

(c) SET NULL

All the values that make up the foreign key are se to NULL if the corresponding row in the parent table is deleted. For this constraint to execute, the foreign ke columns must be nullable.

Foreign key (Did) references departmeot.
- On delete SET NULL
- On Update SET NULL

Insert Null value of 'Did' in the place of deleted 'did

(d) SET DEFAULT

All the values that make up the foreign key are set to their default values if the corresponding row in the parent table is deleted. For this constraint to execute, all foreign key columns must have default definitions

Foreign key (Did) references department
- On delete SET DEFAULT
- On Update SET DEFAULT

1.8. Keys

Page No.
Date

Employee

Id	Name	Aadhar No.	Email_Id	Dept_Id	Tuple
				1	
				2	
				2	
				3	

1) Super Key
Is combination of all possible attributes that can uniquely identify the rows in given relh

i) Superset of candidate key
ii) A table can have many super keys.
iii) It may have additional attribute that are not needed for unique identity

id	Name	Aadhar No.	Id	Dept
A				1
B				2
B				2
V				3

super keys
Emp_Id, Emp_id, Aadhar No., Id, E, A
Adhar No. A, Email Id, Id Name
emailId Empid, -Id

Page No.
Date

Candidate Key

→ It is an attribute or set of attribute which can uniquely identify a tuple

- Minimal Super Key / No redundant attribu
- It not allowed to have NULL values.

candidate Keys

Emp - Id

Aadhar - No.

Primary Key

It is one of the candidate key choosen by the DB designer to uniquely identify tuple in relation

→ i) Not null

ii) Alway Unique (Not duplicate)

iii) can't changed ; No updation is possible.

iv) The value of primary key must be assigned

v) when inserting a record

vi)

Page No.
Date

Foreign Key

- A key is used to link 2 tables together
- An attribute in one table that refers to the primary key in another table

.

Emp Id	Name	Aadhar	Email	Dept
1	A			1
2	N			2
3	N			2
4	V			3

Page No.
Date

DBA

i) It is a person or group that responsible for supervising both the DB & users of DBMs.
ii) DBA coordinate all the activities of the DB system
iii) They use specialized s/w to store & organize data
iv) They take all the permissions.
v) It control whole DB & it is leader.
vi) Task
- schema defination → structure of DB which specified at the time of DB design is called DB schema.
- storage structure & access method def.
- schema & physical org. modification.
- Specifying integrity constraints.
- Granting user authority to access DB
- Monitoring performance & responding to changes in requirements.
- Backup & restoring DB & define security

vii) It is responsible of all 3 levels of DB
viii) DBA is responsible for installation, configuration upgradation, maintenance & monitoring DBMS an org.

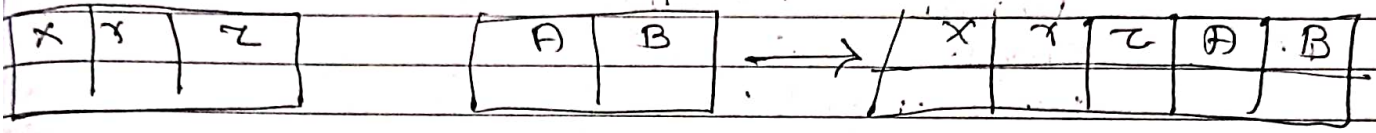
ix) Skill required for DBA is Good communication skills, knowledge of DB architecture & design & RDBMs (oracle, SQL server), knowledge of SQL

DATE / /

SQL Joins

The relational operations can merge columns from two different tables to form new join table.

Table 1 | Table 2 | After joining 1 & 2



we can join multiple tables with the help of some join condition or ~~wing~~ without any join condition.

Syntax →

```
SELECT  column_list
FROM    Table 1, Table 2
WHERE   Join_condn
```

JOIN

1) Equi JOIN (Equal join)
2) NON Equi - JOIN

1) Equi JOIN (Equal JOIN)

Defn → The join query in which equality condition used for joining multiple tables is called as Equal JOINs.

Example :

Employee table

Eid	Ename	Did
1	Mahesh	10
2	Rajesh	20
3	Shiv	40

Department table

D-id	D-name
10	H.R
20	Sales

To join department & Employee table using joining condition that 'did' columns of employee table should match with 'Did' columns of dept table.

```
SELECT  E.Eid [Eid]
        D.Did [Did]
        D.name [DName]
FROM    Employee E,
        Department D
WHERE   E.Eid = D.Did
```

Result →

Eid	Did	Dname
10	10	HR
20	20	sales
30	30	HR

② Non Equi-JOIN

It make use of operators like range operator (BETWEEN) or limit operator (IN) etc.
- Such joins are not used frequently

Defn → The join query in which equality condition is not used then such joins are called as Non Equal JOIN.

e.g → Employee Table

Ename	salary	grade-level
Mahesh	45000	A
Subas	41000	B
Jay	82000	C
Smriti	10000	D
Giri	20000	E
Shubhra	30000	F

Job Grade Table

low-sal	Hig-salary
1	9999
10000	19999
20000	29999
30000	39999
40000	49999
50000	59999

TEJ MY BRAIN
DATE / /

b join dept. & employee table using joining oper.n condh that did columns of

```
SELECT  E.name
        E.salary
        J.grade level
FROM    employees E
        Job grades J
WHERE   E.salary
BETWEEN J.lowest_sal
  AND   J.Highest_sal
```

Result

Employee_job grade Table

Ename	salary	grade_level
Mahesh	45000	E
Suhas	41000	E
Jay	32000	D
Smriti	10000	B
Asiri	20000	C
Shubhra	30000	D

TEJ MY BRAIN
DATE / /

Types of Join

1) Natural join
2) Cartesian product / cross join
3) self join
4) Inner join
5) Outer join.

1) Natural join
- A natural join returns all rows by matching values in common columns having same name & data type of columns & that column should be present in tables.
- It eliminates duplicate column present in JOIN table. Therefore common column will be printed only once in resultant table.
- Joining tables must have at least once common column between them which is also has same column name.
- In this type of join processor looks for a column name which is present in both table than matches data types if match is found than these tables will be joined using common column name

Syntax:-
SELECT column-list
FROM Table 1
NATURAL JOIN
Table 2

Example.
This query automatically select common column & join tables based on that column. no need to specify name of common column explicitly but joining table must have common column with same same & same data type Other wise error will occur.

DATE / /

Employee table

Eid	Ename	Did
1	Mahesh	100
2	Sahas	200
3	Jay	100

Department Table

Did	Dname
100	HR
20	sales

```
SELECT E.Eid [Eid]
       D.Did [Did]
       D.Dname [Dname]
FROM   Employee E
       NATURAL JOIN
       Department D
```

Result →

Eid	Did	Dname
1	100	HR
2	200	Sale
3	100	HR

(2) Cartesian product / cross join

- A cross join performs relational product or cartesian product of 2 tables specified in query
- cross join occur due to WHERE condn is missing in query or some invalid opern in where clause leads to understatized results or CROSS join.

A cartesian product is formed when
- A join condn is omitted
- A join condn is invalid
- All rows in the 1st table are joined to all rows in the 2nd table

- To avoid cartesian product, always include a valid join condition in where clause

TEJ MY BRAIN

DATE / /

Syntax

```
SELECT column list
FROM Table 1
CROSS JOIN
Table 2
```

E.g

This query automatically selects common column & join tables based on that column. no need to specify name of common column explicitly.
- But joining table must have common column with same name otherwise error will occur

Employee table

Eid	EName	Did
1	M	100
2	S	200
3	J	100

Dept. Table

Did	Dname
100	HR
200	Sales

```
select e.Eid CEidJ,
       e.Ename CEnameJ,
       e.Did CDidJ
       d.Did CDidJ
       d.Dname CDnameJ)
FROM Employee e
     CROSS JOIN
     Department d
```

Eid	Ename	Did	Did	Dname
1	M	100	100	HR
1	M	100	200	Sales
2	S	200	100	HR
2	S	200	200	Sales
3	J	100	100	HR
3	J	100	200	Sales

3) Self join

- The tables being joined in a query do not need to distinct, u can join any table to itself as long as you give each table reference a different name using table alias.
- Self joins are useful for discovering relationships between different columns of data in the same table.
- In this type of join query we must use rename oper^n or table alias.

Example

This query joins the Employee table to itself over the Eid & Mid column using table aliases E & M to distinguish the table references.

Employee table (E)

Eid	Ename	Mid
1	M	2
2	G	3
3	S	3

manager table

Eid	Ename	Mid
1	M	2
2	G	3
3	S	3

join E.Mid = M.Eid

Copy to employee table

```
SELECT E.EName [Employee],
       M.Ename [manager]
FROM   Employee E
JOIN
Employee M
ON E.Mid = M.Eid
```

join condn

Employee	manager
M	G
G	S
S	S

TEJ MY BRAIN

DATE / /

Inner join

- In many cases tables are joined according to search condⁿ that find only the rows with matching values this type of join is known as an inner equijoin.
- Inner join joins two table when there is at least one match between two tables.
- Inner join works as simple join

syntax

```
SELECT column-list
FROM Table1
        INNER JOIN
        Table2
ON (JOIN-condⁿ)
```

Example

Employee table(E)

Eid	Ename	Did
1	M	100
2	S	200
3	J	100

Join E.Did

Dept table

Dept-id	Dname
50	HR
100	TIS
200	Sales

D.Dept-id

```
SELECT  E.Eid [Eid],
        C.Did [Did],
        D.Dname [DName]
FROM    Employee E
        INNER JOIN
        Department D
ON  Eid = D.DeptId
```

Eid	Did	DName
1	100	TIS
2	200	Sales
3	100	TIS

DATE / /

5) Outer join

In an inner join or in case of a simple join the resultant table contains only the combinations of rows that satisfy the join conditions. Rows that don't satisfy the join condn are discarded. Outer join joins two table although there is no match between two joining tables.

- An outer join makes one of the tables dominant. Such table is called the outer table & other table is called subordinate table
- The resultant table contains the combinations of rows from dominant table that satisfy the join conditions & also rows that do not having matching rows in the subordinate table.
- The rows from dominant table that do not have matching rows in the subordinate table contain NULL values in the columns selected from subordinate table.

Def ?

TEJ MY BRAIN
DATE / /

iew

i) View is a virtual table based on the result set of an SQL statement

ii) It contains rows & columns, just like real table. The fields in a view are fields from one or more real tables in the database.

iii) A view is a virtual table, which consists of a set of columns from one or more tables. It is similar to a table but it does not store in the dB. view is a query stored as an object.

Syntax → create view <view-name> as select <set of fields> from relation name where (condition)

Updating a view

Syntax → create view view-name as select column-name(s) from table-name where condition.

Dropping a view

A view can be deleted with drop view command

Syntax → Drop view <view name>

DATE / /

Constraints

- Constraints are nothing but limitations or condition imposed on data of database in order to keep database in consistent or correct state.

Example → Employee can have phone no. or cannot phone no. so we can define phone no. attribute as a NULL attribute using NULL constraints.

phone_no char(10) NULL

Syntax

```
CREATE TABLE table_name
(column datatype [column_constraint_Name]
                 [column_constraint],
 column datatype [DEFAULT expr.]
 [column_constraint],
        :
 Table_constraint ] [,...]
)
```

Example

Some attributes in table are not required so such columns can be defined as NULL constraint. In the STUDENT table it is allowed to insert row having phone number as Null.

```
CREATE TABLE STUDENT
(
    Sid varchar(10),
    Name varchar(50)
    Phone_no char(1000) NULL
)
```

DATE / /

Nested subquery

- A subquery is a select from where expression that is nested within another query.
- A common use of subqueries is to perform tests for set membership, make set comparisons & determine set cardinality by nesting subqueries in the where clause.

Page No.:
Date: / /

Set operations

The SQL operations union, intersect & except operate on relations & correspond to the mathematical set theory operations ∪, ∩ & −.

1) Union

i) Union combines two different results obtained by a query into a single result in the form of a table.

ii) Union removes all duplicates if any from the data & only displays distinct values. If the duplicate values are required in the resultant data, then UNION ALL is used.

Syntax

```
Select column name from table name 1
Union
Select column 1 from table name 2
```

Student 1

Id	Name
1	XXX.
2	AAA
3	BBB

Student 2

Id	Name
1	XXX
4	YYY

Example →

```
Select Id, Name from ~~tabl~~ student 1
Union
Select id ~~from~~ , Name from student 2
```

→ Result

Id	Name
1	XXX
2	AAA
3	BBB
4	YYY

2) Union All

Union all are used for if duplicate values are required.

Syntax → select col1, col2 from table 1
Union all
select col1, col2 from table 2

Example → select id, name from student1
Union all
select id, name from student2

Result →

Id	Name
1	XXX
1	XXX
2	AAA
3	BBB
4	YYY

(3) Intersect

i) The intersection operator gives the common data values between the two data sets that are intersected.

ii) The two data sets that are intersected should be similar for intersection operator to work.

iii) Intersection also removes all dupdicates before displaying the result.

Syntax → select col1, col2 from Table 1
Intersect
select col1, col2 from table 2

Example → select id, name from student 1
intersect
select ~~col1~~ id, ~~col2~~ name from student 2

Page No.:
Date: / /

Result →

Id	Name
1	xxx

4) Except / set difference / Minus

The set difference operators takes the two sets & returns the values that are in the first set but not in the second set.

Syntax →

```
select col1, col2 from Table 1
Except
select col1, col2 from Table 2
```

Example →

```
select id, name from ~~Table~~ Student1
Except
select id, name from Student2
```

Result →

Id	Name
2	AAA
3	BBB