UNIT-2 :

* Relational algebra:

" " is a procedual query lang, which takes instances of relations as i/p & yields instances of relations as o/p.

- It uses Operators to perform queries.
- An operator can be either unary or binary.

- select (σ)
- project (π)
- Union ($\cup$)
- Set difference($-$)
- Cartesian prod ($\times$)
- Rename (ρ)

(i). Select: $\overset{\text{sigma}}{\sigma_p(r)}$ It selects tuples that satisfies the predicate.

σ is selection predicate

r is a relation

p is a prepositional logic which may use and, or, not.

- Relational operators like $=, \neq, \geq, \leq, <, >$.

eg: ① $\sigma_{subject = \text{"database"}}(Books)$ || select * from Books where sub="DB"

-which selects the tuples from books where subject is database.

② $\sigma_{subject = \text{"database"} \text{ and } price = \text{"1000"} \text{ or } year > \text{"2010"}}(Books)$

-selects tuples from books where subject is database & price is 1000 or those books published after 2010.

(ii). project ($\overset{\text{pi}}{\pi}$): It projects column's that satisfy given predicate

$\pi_{c_1, c_2, c_3}(r)$; c_1, c_2, c_3 are attribute names of relation r.

- Duplicate rows are automatically eliminated.

eg: $\pi_{subject, author}(Books)$ || select sub, author from Books;

selects & projects colums subject & author from relation books.

(iii). Union operation ($\cup$): select author from Books union select author from Articles;

$r \cup s = \{t \mid t \in r \text{ or } t \in s\}$

- r and s must to have same no of attributes
- Duplicate tuples automatically eliminated.
- Attribute domains must be compatible.

eg: $\pi_{author}(Books) \cup \pi_{author}(Articles)$; — projects name of authors who either written book or an article or both.

(iv). Set difference (-):

r - s

Finds all tuples that are present in 'r' but not in 's'.

Eg: $\Pi_{author}(Books) - \Pi_{author}(Articles)$

- provides names of authors who have written books but not articles.

(v). Cartesian product (X): cross product

Combines info of two different relations into one.

$r \times s = \{qt \mid q \in r \text{ and } t \in s\}$

Eg: Employee

emp_id	ename	edept
1	Smith	A
2	Harry	C
3	John	B

Department

dept_no	dname
A	marketing
B	Sales
C	Legal

Query: Employee X Department

O/p:

emp_id	ename	edept	dept_no	dname
1	Smith	A	A	marketing
1	Smith	A	B	Sales
1	Smith	A	C	Legal
2	Harry	C	A	marketing
2	Harry	C	B	Sales
2	Harry	C	C	legal
3	John	B	A	marketing
3	John	B	B	Sales
3	John	B	C	legal.

(vi). Rename (ρ): rho

- Rename operation is used to rename the O/P relation.

ρ(student1, student)

↳ renaming student relation to student1.

* **Keys:** Key is an attribute or set of attributes that uniquely identifies a tuple in a relation.

(i). primary key:

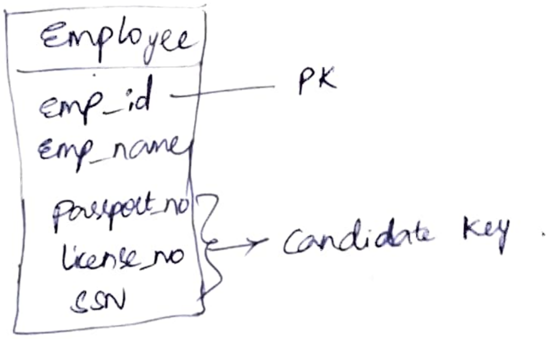
- Used to identify only one instance of an entity uniquely.

Eg: emp_id is PK of Employee entity.

(ii). Candidate key:

- Is an attribute or set of attributes that can uniquely identifying a tuple. There can be only one PK in atable. Won't allow null values.
- ~~except primary key, remaining attributes are considered as Candidate key. the candidate keys are as strong as primary key.~~

Eg: In Employee table,



these no's are different for all.

(iii). Super Key:

- Is an attribute set that can uniquely identifying a tuple.
- Super key is a superset of ~~candidate~~ primary key.

Eg: (Employee_id, emp_name)

(iv). Foreign key:

- " " are the column of the table used to point the PK of another table.

* vi Integrity Constraints:

- " " are set of rules used to maintain the quality of information.

- " " used to protect against any damage to the DB.

types:

i) Domain Constraints:

- " " can be defined as the definition of a valid set of values for an attribute.

- Data type for domain includes string, character, integer, time, date, ...

Eg: Student

RNo	Name	Semester	Age
1	Rama	4th	20
2	Sita	4th	21
3	Ammu	2nd	A

A → Not allowed. Age is an integer attribute.

ii) Entity Integrity Constraints:

- " " " states that primary key value can't be null. Because PK value used to identify individual rows in relation.

- A table can contain a null value other than PK field.

Eg: employee

EmpId	Name	Salary
1	Rama	5000
2	Sita	10000
□	Ammu	6000

Not allowed. PK can't have null.

iii) Referential Integrity Constraints:

- " " " specified b/w 2 tables.

- If FK in Table1 refers to PK of Table2, then every value of the FK in Table1 must be available in Table2 or null.

Eg: Table 1

FK

Emp_id	name	age	d_no
1	Rama	20	11
2	Sita	21	18
3	Ammu	22	22

18 → 18 is not allowed Because it is not present in D_no in table2.

Table 2

PK

D_no	D_location
11	Delhi
22	Hyd
33	mumbai

iii. Key Constraints:

- Keys are used to identify an entity with in the entity set uniquely.
- PK value should be unique & not null.

* **Views:** Uses(Restricting data access, Hides data complexity, Store complex queries, multiple view) _(Rename columns, Simplify commands)

" Are considered as virtual table, contains rows & columns.

- To create a view, we can select the fields from one or more tables present in DB.
- A view can contain either all the rows of a table or only few rows based upon certain condition.

Eg: Student

RNo	Name	Address
1	Rama	Hyd
2	Sita	Delhi
3	Ammu	Hyd

Marks

RNo	Name	marks
1	Rama	99
2	Sita	99
3	Ammu	97
4	Raju	95

- Creating a View: Syntax:

- Create View view_name as
  Select column1, column2, ... from table name where condition.

Eg: - Create view detailsview as
  select name, address from student where RNo < 3;

- select * from details_view;

O/p:

name	address
Rama	Hyd
Sita	Delhi

- Creating a view from multiple tables

Syntax: Create View marksview as
  select student.name, student.address, marks.marks from student, marks
  where student.name = marks.name;

- select * from marksview;

O/p:

name	Address	marks
Rama	Hyd	99
Sita	Delhi	99
Ammu	Hyd	97

- Deleting a view:
  drop view view_name;
- Inserting a row in a view:
  insert into view_name (column1, column2, ...) values(value1, value2, ...);
- Deleting a row: Delete from view_name where condition;
  - Update view; To add or remove fields; Create or replace

Unit-2-(3)

T1:

RNO	Name
1	Ammu
2	Rama
~~2~~	~~Rama~~
3	Sita

T2:

RNO	Name
2	Rama
4	Raja
5	Rani

* **Inner join:** return the recards where intersect.

select table1.rno, table1.name from table1 inner join table2 on table1.rno = table2.rno;

O/P:

RNO	Name
2	Rama

* **Left Outer join:**

select table1.rno, table1.name,
table2.rno, table2.name
from table1 left join table2 on table1.rno = table2.rno;

O/P:

T1		T2	
RNO	Name	RNO_1	Name_1
2	Rama	2	Rama
3	Sita	null	null
1	Ammu	null	null

* **Right Outer join:**

O/P:

T1		T2	
R.NO	Name	RNO_1	Name_1
2	Rama	2	Rama
null	null	5	Rani
null	null	4	Raja

* **Full join:**

O/P:

T1		T2	
RNO	Name	RNO_1	Name_1
2	Rama	2	Rama
null	null	4	Raja
null	null	5	Rani
3	Sita	null	null
1	Ammu	null	null

* Relational Calculus:

" " is non procedural query lang.

- " , is not same as differential & integral calculus but it takes its name from a branch of symbolic logic i.e., predicate calculus

(i). Tuple relational Calculus:

" " " specifies to select the tuples in a relation. It can select the tuples with range of values or tuples for certain attribute values. The resulting relation can have one or more tuples.

Syntax: $\{T \mid P(T)\}$ or $\{T \mid Condition(T)\}$

T is resulting tuples & P(T) is condition used to fetch T.

Eg: $\{T \mid Employee(T)$ and $T.dept\_id = 10\}$

It selects all the tuples of employee name who work for dept 10.

(ii). Domain relational calculus:

" " " uses list of attributes to be selected from the relation based on the condition.

Syntax: $\{a_1, a_2, a_3, \ldots, a_n \mid P(a_1, a_2, a_3, \ldots, a_n)\}$

where $a_1, a_2, a_3 \ldots a_n$ are attributes of the relation.

P is the condition

Eg: $\{ \mid <Employee>\ dept\_id = 10\}$

It selects Emp_id and Emp_name of employees who work for department 10.

* SQL - Null values:

- " " is used to represent a missing value. A Null value in a table is a value in a field that appears to be blank.

Syntax Eg: Create table ~~student~~ Employee (

```
        sno primary key int not null,
        name varchar(10) not null,
        salary number (8, 2));
```

- Here, not null represents the column value should not be null.
- Salary attribute value could be null.

> select sno, name, salary from Employee where salary is not null;

- It will give the records where salary value is not null;

*. Index Definition in SQL:

Index is a schema object. It is used by the server to speed up the retrieval of rows.

Syntax: create index index on table column; /* for single column */

- create index index on table (column1, column2, ---); /* multiple colum* /* composite */

unique indexes:

" " are used for the maintenance of the integrity of the data present in the table as well as for fast performance

It will not allow to enter multiple values into the table

- create unique index index on table column;

When to create:

- A column contain wide range of values
- " " doesn't contain large no. of null values.
- one or more columns are frequently used together in where clause or in join.

- Drop index index;
- alter index indexname on tablename rebuild;