DBMS

MODULE 3
Relational Model and Relational Algebra

Written & Compiled by,
Prof.Priyanka Bhilare

3. Relational Model and relational algebra.

(*) Introduction to Relational Model.

Data Model:-

It is basically a collection of concepts that can be used to describe the structure of the database.

It provides specification describing how database is structured & used.

Following are the examples of data model.

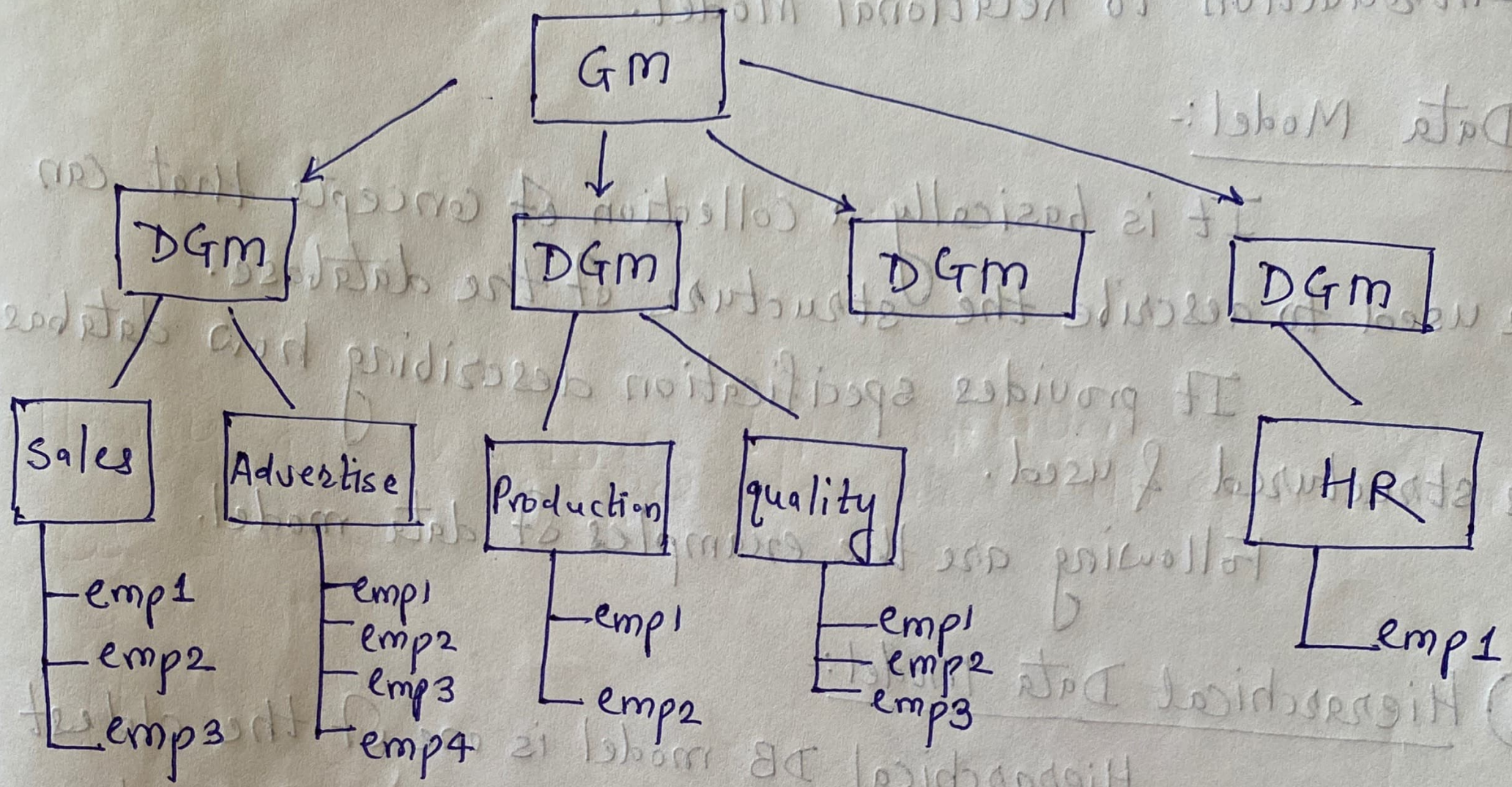
(A) Hierarchical Data Model:-

Hierarchical DB model is one of the oldest database model. This model is like structure of the tree with the records forming the nodes and fields forming the branches of the tree.

Consider, a company's organizational structure at the top we have general manager (GM), under him we have several deputy general manager (DGM). Each DGM looks after a department. Each dept. will have manager & many employees.

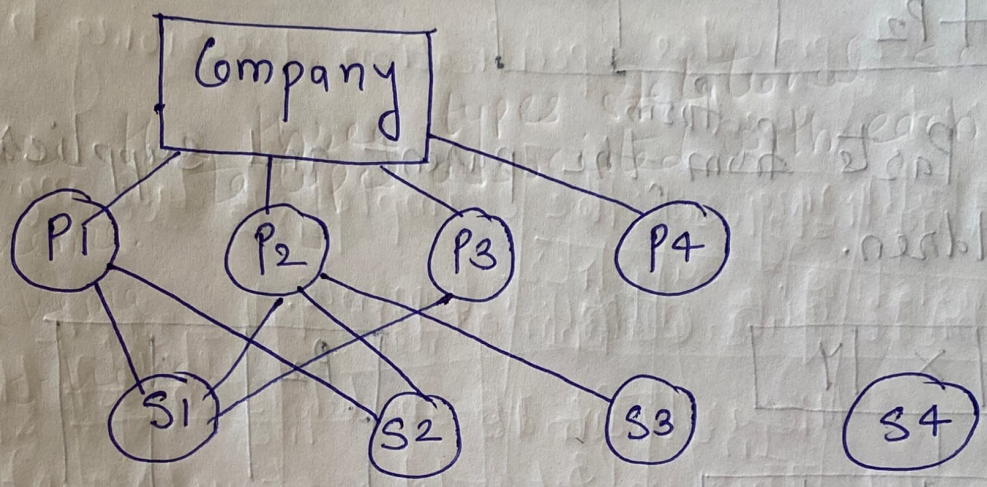
A tree structure may establish (one to many) relationship.

Following figure illustrates the company's organization.



(B) Network Model :-

This is an extension to the hierarchical model. In this data model, data is organized like a graph and are allowed to have more than one parent node. (defines 1:N, 1:1, M:N relationship)



This was most widely used DB model, before relational model was introduced.

(C) Entity-Relationship Model :-

It describes the relationship between different entities. It is basically graphical approach used to define conceptual view of DB.

Lets take eg. of college management system in which Student and teacher are two entities.

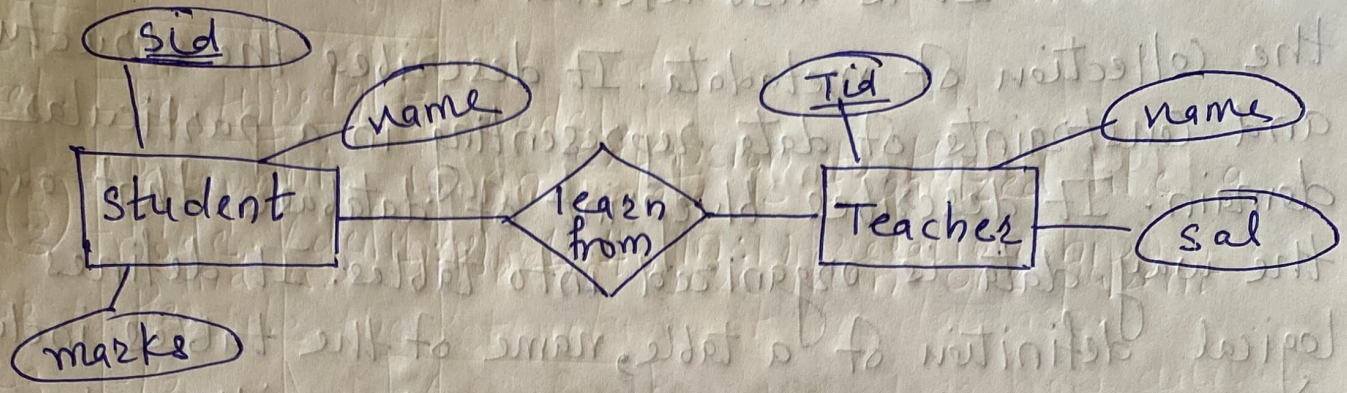


fig. E-R model

(D) Relational model :-

In this model ~~is~~ data is organized in 2D tables and relationship is maintained by storing a common field.

The basic structure of data in Relational model is table. All the information related to a particular type is stored in rows of that table. Hence, tables are also known as relations in relational model.

Student

sid	name	marks

learnfrom

sid	Tid

Teacher

Tid	name	sal

(*) Relational Schema :-

It is also referred to as DB schema. It is the collection of metadata. It describes the structure and constraints of data representing in a particular domain. It describes blueprint of database that outlines the way data is organized into tables. It defines logical definition of a table, name of the tables including column names and data types associated with each column.

Following are the three types of relational schema.

i) Logical schema.
ii) physical schema
iii) Conceptual schema.

Rules for mapping ER/EER diagram into relational model.

1) Entity :-

i) For strong entity create separate table.

ii) For weak entity create separate table and primary key of weak entity will be primary key of strong entity + any column of weak entity.

3) Attribute :-

i) For composite attribute don't create separate table.

ii) For multivalued attribute create separate table.

iii) For descriptive attribute create separate table.

Note:- If descriptive attribute is not present then whether to create separate table or not is decided by the cardinality exists between two entities.

) Relationship :-

i) If there is 1:1, 1:m, m:1 relationship exists between two entities then 1's primary key is added in another table.

ii) If there is m:N relationship exists between two entities then create separate table and primary key of that table will be primary keys of

both the tables between which M:N relationship exists.

d) Inheritance:-

Create separate table for subentities and subentities will include primary key of parent entity. plus any column of subentity.

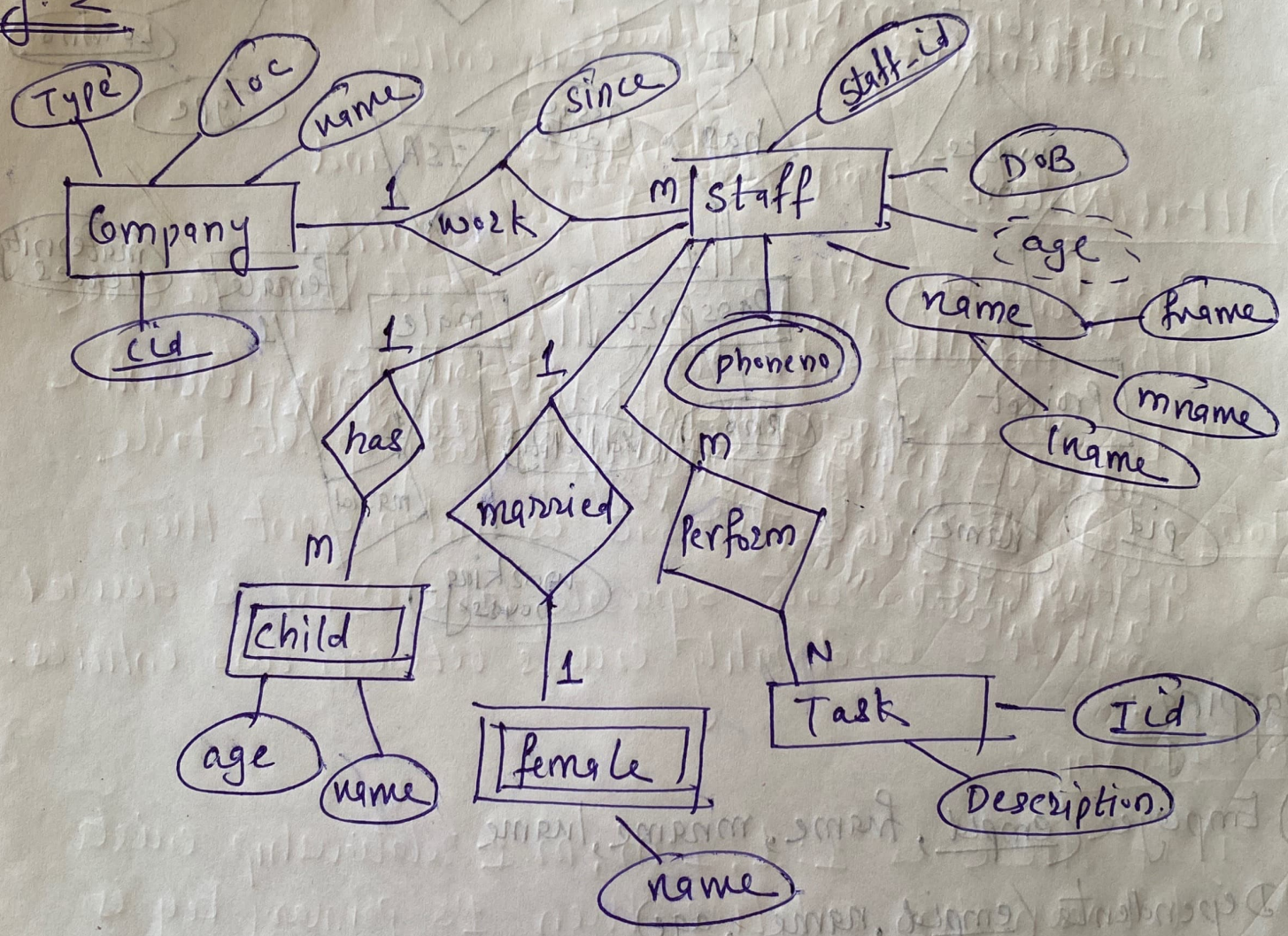
Note:-

1) Primary key of multivalued attribute will be primary key of the entity to which multivalued attribute belongs + multivalued attribute itself.

2) Primary key of descriptive attribute will be the primary keys of the entities betn ~~with~~ which descriptive attribute exists + any attribute associated with descriptive attribute.

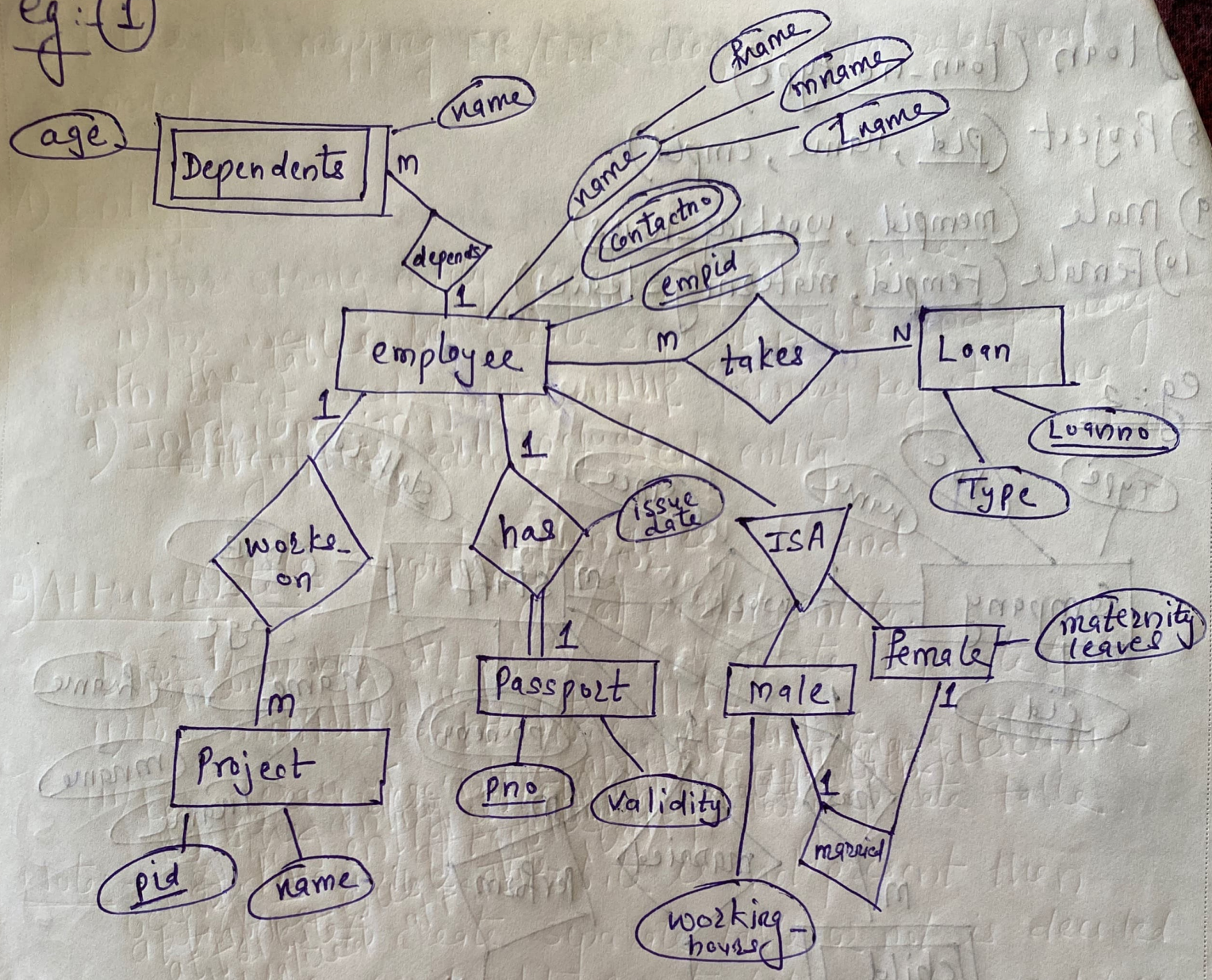
7) loan (loan_no, type)
8) Project (pid, name, empid)
9) Male (mempid, working_hours)
10) Female (Fempid, maternity_leave)

eg:- 2



1) Company (cid, type, loc, name
2) work (cid, staff_id, since)
3) staff (staffid, DoB, fname, mname, lname
4) child (staffid, name, age)

eg:- (1)



Mapping :-

1) Employee (empid, fname, mname, lname
2) Dependents (empid, name, age)
3) emp_contact (empid, emp_contactno)
4) takes (empid, loanno)
5) has (empid, pno, issue_date)
6) Passport (pno, validity)

5) female (staff_id, name)

6) staff_contact (staff_id, phone no)

7) Perform (staff_id, Tid)

8) Task (Tid, description)

Relational Algebra (RA)

Relational Algebra is a procedural query language used to query DB tables to access data in diff^n ways.

In RA input is a relation (table from which data has to be accessed) and output is also relation (a temporary table holding the data asked by the user).

Consider foll. eg.

id	name	age
1	A	17
2	B	19
3	C	15
4	D	13

fig. student table

we use RA, to fetch data from this table

select name from the student table with age less than 17.

o/p

name
C
D

← o/p is also in the form of the table which results in diff^n columns.

The primary operations that we can perform using relational algebra are :-

Unary operators :- (as they operate on one rel^n)

1) select (σ) sigma
2) Project (π) pi
3) rename (ρ) rho

Binary operators :- (as they operate on pairs of the rel^n)

1) Join ($\bowtie$) bow-tie
2) Product ($\times$) cross
3) Union ($\cup$) cup
4) Intersection ($\cap$) cap
5) difference ($-$) minus

1) Select operator :-

Select is used to select subset of tuples of the relation that satisfies select condition. It is denoted by lowercase greek letter sigma (σ)

empid	name	job	sal	deptno
1	x	HR	17000	30
2	y	analyst	20000	20
3	z	Tester	15000	20

emp table

deptid	Dept name	Loc
10	store	Mum
20	research	Pune
30	Testing	Kol
40	Marketing	Pune

Dept table

Syntax :-

$\boxed{\sigma_{(condition)} (relation\ name)}$

i) Find all the emp. working in dept 20.

The sql query is,

select * from emp where deptid=20;

The equivalent relational algebra is,

eg :- $\boxed{\sigma_{(deptid=20)} (emp)}$

we use relational operators <, >, <=, >=, !, <> and Logical operators and ($\wedge$), OR ($\vee$) and not ($\neg$) conditions can be combined together using operators.

ii) Find employees working in dept 20 and name as y.

The equivalent RA query is,

$\sigma_{(deptid=20 \wedge name='y')} (emp)$

iii) Find employees having salary greater than 15000.

$\sigma_{(sal>15000)} (emp)$

2) Project operation :-

Project operation is used to select attributes (columns) that satisfy the given condition.

Syntax :- $\pi_{column1, column2 \ldots column\,n}$ (relation name)

<u>Note</u> :- Selection operator only selects the required tuples but doesnot display them. For displaying data projection operator is used.

i) display all employee id and their names.

$\pi_{empid, name}$ (emp)

select and project can be combined together.

2) Display id & name of the employees working in the department 20 or 30.

$\pi_{id, name}$ ($\sigma_{dept\,id=20 \text{ or } dept\,id=30}$) (emp)

3) Deduct 2000 from the salary of all the employees.

$\pi_{name, sal-2000}$ (emp)

3) Rename operator :- (ρ)

The rename operator returns an existing relation under a new name

a) $\rho_A(B)$ — relation B changed its name to relation A.

eg:- $\rho_{employee}(emp)$ - emp relation renamed to employee.

b) $\rho_{A(new_attribute_names)}(B)$ - It renames both relation and all the attributes of relation.

c) $\rho_{(new_attribute_names)}(B)$ - renames only the attribute names.

d) $\rho_{fname/name}(emp)$ - only few attributes are renamed. (name attribute renamed to fname)

(*) Division operator :-

Division operator $A \div B$ can be applied if and only if

1) Attribute of B is proper subset of attribute of A.
2) The relation returned by division operator will have attributes = (All attributes of A − All attributes of B).
3) The relation returned by division operator will return those tuples from relation A which are associated with every B's tuple.

eg. (1)

empid	Productname
1	Shirt
1	Jeans
2	Top
2	Tshirt
4	Jeans
4	Tshirts

emp table

Productname
Top
Tshirt

Product table.

To apply division operator as, emp ÷ product

1) The operation is valid as attributes in product is a proper subset of attributes in emp.

2) The attributes in resulting relation will have attributes {empid, Productname} − {Productname} = {empid}.

3) Tuples in resulting relation will have those empid which are associated with all B's tuple. {Top, Tshirt} empid 1, 4 are associated to shirt and Jeans only, whereas empid 2 is associated with all the tuples of B. so, resulting relation will be,

o/p

empid
2

eg (2) course taken by the students,

student table

student_name	course name
Ashish	DBMS
Ashish	OOPM
Komal	DBMS
Komal	OS
Trushali	OOPM
Trushali	ML
Riya	OS

course name
DBMS
OS

course table

* Set operations :-

1) Union :-

The union of two relations includes all the tuples that either in 'A' or in 'B' or in both 'A' and 'B'. Duplicate tuples are eliminated.

a) both 'A' and 'B' must have same no. of columns.

b) Domain of each attribute in column order is the same in both 'A' and 'B'.

Syntax:- $\pi_{\text{columnname}(s)}(A) \cup \pi_{\text{columnname}(s)}(B)$

$\pi_{\text{name,number}}(R) \cup \pi_{\text{name,number}}(S)$

R

name	number
A	1
B	2
C	3
F	4
E	5

S

name	number
A	1
C	2
D	3
E	4

o/p R ∪ S

name	number
A	1
B	2
C	3
F	4
E	5
C	2
D	3
E	4

2) Intersection :-

The intersection of 'R' and 'S' is a relation that includes all the tuples that are in both R & S.

<u>Syntax</u>:- $\pi_{columname(s)}(R) \cap \pi_{columnname(s)}(S)$

$\pi_{name,number}(R) \cap \pi_{name,number}(S)$

<u>o/p</u>

name	number
A	1

3) Difference (Minus) :-

The difference of R & S is the relation that contains all the tuples of R which are not in S.

<u>syntax</u>:-

$\pi_{columnames}(R) - \pi_{columnnames}(S)$

<u>o/p</u> :-

$\pi_{name,number}(R) - \pi_{name,number}(S)$

name	number
B	2
C	3
F	4
E	5

Joins in Relational Algebra :-

A join operation combines related tuples from different relations if and only if a given join condition is satisfied. It is denoted by ⋈.

1) Theta Join :- (θ)

Theta join combines tuples from different relations provided they satisfy the theta condition. The theta join (θ) condition is denoted by symbol θ.

Syntax :-

$$A \bowtie_{\theta} B$$

where A, B are relations.

eg:-

emp_code	Salary
101	50000
102	30000
103	25000

emp

emp_code	ename	salary
101	Priyanka	50000
102	Snehal	30000
103	Naina	25000

salary_info

emp ⋈ emp.emp_code > salary_info.emp_code salary_info

o/p

emp_code	salary	emp_code	ename	salary
102	50000	101	Priyanka	50000
103	30000	101	Priyanka	50000
103	25000	102	Snehal	30000
103	25000	103	Naina	25000

2) Equijoin :-

When theta join uses only ~~equalitity~~ equality comparison operator, it is said to be equijoin

eg:-

emp ⋈ emp.emp_code = salary_info.emp_code salary_info.

O/P

emp_code	salary	emp_code	ename	salary
101	50000	101	Priyanka	50000
102	30000	102	Snehal	30000
103	250000	103	Naina	25000

3) Natural Join :-

Natural join doesnot use any comparison operator. It does not concatenate the way a cartesian product does. We can perform a natural join if there is at least one common attribute that exists between two relations. In addition, the attributes must have the same name & domain.

eg:- emp ⋈ salary_info

O/P

emp_code	salary	ename	salary
101	50000	Priyanka	50000
102	30000	Snehal	30000
103	25000	Naina	25000

4) Outer Join :-

Outer join is basically of three types.

a) Left outer join
b) Right outer join
c) Full outer join

a) Left outer Join :-

All the tuples from the left relation, R are included in the resulting relation. If there are tuples in R without any matching tuple in the right relation S, then the S-attributes of the resulting relation are made null.

It is denoted by ⟕

eg:- R ⟕ B

emp ⟕ salary_info.

emp (FK)	Branch	salary (PK)
Ram	Infosys	10000
shyam	Wipro	20000
kuber	HCL	30000
hari	TCS	50000

Salary_info.

emp (PK)	street	city
Ram	civil line	mumbai
shyam	Park street	kolkata
Ravi	M.G. road	Delhi
hari	Nehru Nagar	Pune

emp

o/p

emp	street	city	branch	salary
Ram	civil line	mumbai	Infosys	10000
Shyam	Park street	kolkata	Wipro	20000
Ravi	MG Road	Delhi	NULL	NULL
hari	Nehru Nagar	Pune	TCS	50000

b) Right Outer Join :-

All the tuples from the right relation. S are included in the resulting relation. If there are tuples in S without any matching tuple in R then the R attributes of resulting relation are made NULL.

eg:- emp ⟖ salary_info.

emp	Street	city	branch	Salary
Ram	civil line	mumbai	Infosys	10000
Shyam	~~Wipro~~ Park street	kolkata	wipro	20000
kuber	null	null	HCL	30000
Hari	Nehru nagar	Pune	TCS	50000

(c) Full Outer Join :-

All the tuples from both participating relations are included in the resulting relation. If there are no matching tuples for both relations, their respective unmatched attributes are made null.

eg. emp ⟗ salary_info

emp	street	city	branch	salary
Ram	Civil lines	mumbai	infosys	10000
shyam	Park street	kolkata	Wipro	20000
Ravi	MG Road	Delhi	null	null
hari	Nehru Nagar	Pune	TCS	50000
kuber	NULL	NULL	HCL	30000