**RELATIONAL ALGEBRA**

Relational Algebra is **procedural query language**, which **takes Relation** as input and **generate relation** as output.

# Projection (π)

Projection is used to **project required column** data from a relation.

**R**

(A B C)

----------

1. 2 4
2. 2 3
3. 2 3
4. 3 4

**Π b, c (R)**

B C

----

2 4

* 1. 3
  2. 4 \*by default projection **removes duplicate data.**

# Selection (σ)

Selection is used to **select required tuples** of the relations. **π (σ (c>3) R)**

**(**Select tuples from R where c is greater than 3**)**

A B C

-----

1 2 4

4 3 4

# Cross Product (X)

Cross product between two relations A and B (A X B) will result **all the attributes of A followed by each attribute of B**. Each record of A will pair with every record of B.

A B

(Name Age Sex ) (Id Course)

------------------ -------------

Ram 14 M 1 DS

Sona 15 F 2 DBMS

Kim 20 M

A X B

Name Age Sex Id Course

---------------------------------

Ram 14 M 1 DS

Ram 14 M 2 DBMS

Sona 15 F 1 DS

Sona 15 F 2 DBMS Kim 20 M 1 DS

Kim 20 M 2 DBMS

\*If A has ‘n’ tuples and B has ‘m’ tuples then A X B will have ‘n\*m’ tuples.

# Natural Join ()

Natural join is a binary operator. Natural join between two or more relations will result set of all combination of tuples **where they have equal common attribute**.

Emp Dep

(Name Id Dept\_name ) (Dept\_name Manager)

------------------------ ---------------------

1. 120 IT Sale Y
2. 125 HR Prod Z
3. 110 Sale IT A
4. 111 IT

Emp  Dep

Name Id Dept\_name Manager

-------------------------------

A 120 IT A

1. 110 Sale Y
2. 111 IT A

# Rename (ρ)

Rename is a unary operation used for renaming attributes of a relation. ρ (a/b) R will **rename the attribute ‘b’** of relation **by ‘a’**.

**Query-1: find name of all the person who had taken a loan from bank**

CUSTOMER

Columns -> customerID, name, address

LOAN

Columns -> loanID, customerID, amount

* Π (name) (CUSTOMER  LOAN)
* Π (name)(σ(CUSTOMER.customerID = LOAN.customerID)(CUSTOMER X LOAN))

**Query-2: find the customer id who has taken a loan as well as deposited to savings.**

DEPOSIT

Columns-> depositID, customerID, amount

* Π (customerID)(LOAN)

∪

Π (customerID)(DEPOSIT)

***Assignment (←)***

It assigns a value / table / scalar / vector to a variable

\*Scalar is when there is a single result (i.e., no multiple columns and row)

\*Vector is opposite of the scalar.

EMP => columns -> empid, name, sal, did

DEPT => columns -> deptid, dname, dloc

**Query-3: find all the name of the employee who are in finance dept and salary greater than 15000**

**π** (σ **(**dname = “finance” ∩ sal > 15000**)(**EMP  DEPT))

**Grouping(G)**

Grouping can be done by using (fieldname)G (aggregate operation)

Some aggregate functions are

AVG,SUM,MAX,MIN,COUNT..

Query-4: Average salary of employees

G\_insuffix(AVG(sal)) (EMP)

Query-5: