Relational Algebra in DBMS

1. Introduction to Relational Algebra

Relational Algebra is a **procedural language** used to **query relational databases**. It provides a set of **operations** to manipulate **relations (tables)** and retrieve the required data.

Introduced by: E.F. Codd (1970)
Purpose: Defines how data can be retrieved rather than specifying the actual steps.
Used in: SQL query optimization and **database management systems (DBMS)**.

---

2. Types of Relational Algebra Operations

Relational Algebra operations are classified into two categories:

Basic (Fundamental) Operations

- Selection (σ)
- Projection (π)
- Union (∪)
- Set Difference (-)
- Cartesian Product (×)
- Rename (ρ)

Advanced (Derived) Operations

- Intersection (∩)
- Join Operations
- Division (÷)

---

3. Fundamental Operations of Relational Algebra

Selection(σ) - Filtering Rows

The **Selection (σ)** operation is used to **extract specific rows (tuples) from a relation (table) based on a condition**.

Symbol: σ condition (Relation)

Example Table: STUDENT

Roll No	Name	Dept	Marks
1	Raju	IT	78
2	Ravi	ECE	65
3	Rani	IT	82
4	Sita	CSE	59

Query: Students from IT with Marks ≥ 80

Relational Algebra: σ Dept = 'CSE' ∧ Marks ≥ 80 (STUDENT)

Output:

Roll	Name	Dept	Marks
3	Rani	IT	82

Projection (π) – Selecting Columns

The **Projection (π)** operation is used to **extract specific columns (attributes) from a table**.

Symbol: π attribute_list (Relation)

Query: Find only **Student Names** from the STUDENT table:

Relational Algebra:
π **Name** (STUDENT)

Output:

Name
Raju
Ravi
Rani
Sita

Union (∪) – Combining Rows from Two Tables

The **Union (∪)** operation **combines** data from two relations and removes duplicates.

Condition:

- Both tables must have the **same number of columns**.
- Columns must have the **same data type**.

EMP1 Table:

EmpID	Name	Dept
101	Ajay	IT
102	Bala	HR

EMP2 Table:

EmpID	Name	Dept
102	Bala	HR

103	Ravi	IT

Query: Find all employees who are present in either EMP1 or EMP2.

Relational Algebra: EMP1 ∪ EMP2
Output:

EmpID	Name	Dept
101	Ajay	IT
102	Bala	HR
103	Ravi	IT

Set Difference (-) – Finding Unique Rows

The **Set Difference (-)** operation **finds records in one table but not in another**.

Example: Find employees who are in EMP1 but NOT in EMP2.

Relational Algebra:
EMP1 – EMP2

Output:

EmpID	**Name**	**Dept**
101	**Ajay**	**IT**

Cartesian Product (×) – Combining Tables Without a Condition

The **Cartesian Product (×)** operation combines **each row of Table A** with **each row of Table B**, producing all possible combinations of rows.

Relational Algebra:
EMP × DEPT

EMP

EmpID	**Name**
101	Ajay
102	Bala

DEPT

DeptID	**DeptName**
1	IT
2	HR

Query:
Find all possible combinations of employees and departments.

Relation Algebra: EMP × DEPT

Output:

EMP × DEPT produces:

EmpID	Name	DeptID	DeptName
101	Ajay	1	IT
101	Ajay	2	HR
102	Bala	1	IT
102	Bala	2	HR

Note: Cartesian Product is rarely used directly. It is mostly used in **Join Operations**.

Rename (ρ) – Changing Table or Column Names

The **Rename (ρ)** operation is used to **change the name of a relation or its attributes**.

1. Rename the Relation (Table)

Query:
Rename STUDENT table as S.

Relational Algebra:
ρ S (STUDENT)

2. Rename the Attributes (Columns)

Query:
Rename the attributes of STUDENT as NewRoll, NewName, NewDept.

Relational Algebra:
ρ S(NewRoll, NewName, NewDept) (STUDENT)

Rename the attributes of STUDENT table as:

- **Roll → NewRoll**
- **Name → NewName**
- **Dept → NewDept**

Now, the table is S instead of STUDENT.

4. Advanced Operations of Relational Algebra

Intersection (∩) – Common Rows Between Two Tables

The **Intersection (∩)** operation **returns only the common tuples** in two relations.

Query:

Find employees who are present in **both** EMP1 and EMP2.

Relational Algebra:

EMP1 ∩ EMP2

EMP1

EmpID	Name	Dept
101	Ajay	IT
102	Bala	HR
103	Ravi	IT

EMP2

EmpID	Name	Dept
102	Bala	HR
103	Ravi	IT
104	Meera	CSE

Output (Common Rows Only):

EmpID	Name	Dept
102	Bala	HR
103	Ravi	IT

Join Operations:

Why Join Operation Gets Special Attention?

Join gets special attention because it is the main operation used to combine related data from multiple tables in a relational database. Since data is stored separately through normalization, most queries require joining tables using common attributes (like foreign keys). Hence, join becomes an essential and frequently used operation.

Join Operations – Combining Tables Based on Conditions

Joins combine tables **based on a common attribute**.

Types of Joins in Relational Algebra:

- **Theta Join (⋈ condition)** → Uses a general condition.
- **Equi Join (⋈ attribute = attribute)** → Uses equality condition.
- **Natural Join (⋈)** → Removes duplicate attributes.
- **Outer Joins** → Includes unmatched rows (Left, Right, Full).

EMP

EmpID	Name	DeptID
101	Ajay	1
102	Bala	2
103	Ravi	1
104	Sita	3

DEPT

DeptID	DeptName
1	IT
2	HR
4	Finance

1. Theta Join (⋈ condition)

Join using **any general condition** (>, <, =, ≤, etc.)

Query:

Combine EMP and DEPT where EMP.DeptID = DEPT.DeptID.

Relational Algebra:

EMP ⋈ **(EMP.DeptID = DEPT.DeptID)** DEPT

Output:

EmpID	Name	DeptID	DeptID	DeptName
101	Ajay	1	1	IT
102	Bala	2	2	HR
103	Ravi	1	1	IT

(DeptID 3 in EMP and 4 in DEPT do NOT match)

2. Equi Join (⋈ attribute = attribute)

A special type of Theta Join that uses **only equality (=)**.

Query:

Join EMP and DEPT where DeptID is equal.

Relational Algebra:

EMP ⋈ **(EMP.DeptID = DEPT.DeptID)** DEPT

Output:

(Same as theta join because condition is equality)

EmpID	Name	DeptID	DeptID	DeptName
101	Ajay	1	1	IT
102	Bala	2	2	HR
103	Ravi	1	1	IT

3. Natural Join (⋈)

Automatically joins tables **on common attribute names** and **removes duplicate columns**.

Query:

Natural join EMP and DEPT.

Relational Algebra:

EMP ⋈ DEPT

Output:

Duplicate DeptID is removed.

EmpID	Name	DeptID	DeptName
101	Ajay	1	IT
102	Bala	2	HR
103	Ravi	1	IT

4. Outer Joins – Includes unmatched rows

4.1 Left Outer Join (⟕)

Returns **all rows from the left table** (EMP) and matching rows from DEPT.
Unmatched rows get **NULLs**.

Query:

Get all employees and their departments (even if no match).

Relational Algebra:

EMP ⟕ DEPT

Output:

EmpID	Name	DeptID	DeptName
101	Ajay	1	IT
102	Bala	2	HR
103	Ravi	1	IT
104	Sita	3	**NULL**

(Sita has no matching department)

4.2 Right Outer Join (⟖)

Returns **all rows from DEPT**, and matching rows from EMP.

Query:

Get all departments with employees (even if no employee exists).

Relational Algebra:

EMP ⟖ DEPT

Output:

EmpID	Name	DeptID	DeptName
101	Ajay	1	IT
102	Bala	2	HR
103	Ravi	1	IT
NULL	**NULL**	4	Finance

(Finance has no employees)

4.3 Full Outer Join (⟗)

Returns **all rows from both tables**, unmatched rows get NULLs.

Query:

Show all employees and all departments, even if no match exists.

Relational Algebra:

EMP ⟗ DEPT

Output:

EmpID	Name	DeptID	DeptName
101	Ajay	1	IT
102	Bala	2	HR
103	Ravi	1	IT

EmpID	Name	DeptID	DeptName
104	Sita	3	**NULL**
NULL	**NULL**	4	Finance

(Includes unmatched rows from both sides)

Division (÷) – Finding Related Data

The **Division (÷)** operation is used when we need to **find entities that are related to all values in another set**.

TABLE: ENROLLS (Student → Course)

(Which student took which course)

Student	Course
Raju	DBMS
Raju	SQL
Ravi	DBMS
Rani	DBMS
Rani	SQL

TABLE: REQUIRED_COURSES

(Courses that must be completed)

Course
DBMS
SQL

Query:

Find students who have taken **all** required courses.

Relational Algebra:

ENROLLS ÷ REQUIRED_COURSES

Output:

Student
Raju
Rani

Relational Calculus – A Non-Procedural Query Language

Relational Calculus is a **non-procedural** query language.
This means:

- You specify **WHAT** result you want
- You do **not** specify **HOW** to get it

The DBMS decides the steps internally.

Relational Calculus focuses on **describing the result**, not the process.

✔ **Types of Relational Calculus**

1. **Tuple Relational Calculus (TRC)**
2. **Domain Relational Calculus (DRC)**

1. Tuple Relational Calculus (TRC)

In TRC, we write **conditions on tuples** (rows).

General Form:

{ t | condition on t }

Example Table: STUDENT

Roll	Name	Dept	Marks
1	Raju	IT	78
2	Ravi	ECE	65
3	Rani	IT	82
4	Sita	CSE	59

Query:

Find names of students who are in the IT department.

TRC Expression:

{ t.Name | t ∈ STUDENT AND t.Dept = "IT" }

Output:

Raju
Rani

2. Domain Relational Calculus (DRC)

In DRC, we write **conditions on individual attribute values** (domains).

General Form:

{ <x_1, x_2, …> | condition on values }

Example (same query):

Find names of IT students.

DRC Expression:

{ x | ∃r, m (STUDENT(x, r, "IT", m)) }

(Meaning: select name **x** where there exists a tuple with Dept = "IT")

Output:

Raju
Rani