What is Relational Algebra?

The relational algebra is a theoretical procedural query language which takes an instance of relations and does operations that work on one or more relations to describe another relation without altering the original relation(s). Thus, both the operands and the outputs are relations.

The primary operations of relational algebra are as follows:

- Select
- Project
- Union
- Set different
- Cartesian product
- Rename

Select Operation

It selects tuples that satisfy the given predicate from a relation.

Notation − σp(r)

Here σ stands for selection predicate, and r stands for relation, and p is a propositional logic formula which may use connectors like and, or, and not.

σ predicate(R): This selection operation functions on a single relation R and describes a relation that contains only those tuples of R that satisfy the specified condition (predicate).

Roll	Name	Department	Fees	Team
1	Bikash	CSE	22000	A
2	Josh	CSE	34000	A
3	Kevin	ECE	36000	C
4	Ben	ECE	56000	D

Select all the student of Team A :

```
σ Team = 'A' (Student)
```

Roll	Name	Department	Fees	Team
1	Bikash	CSE	22000	A
2	Josh	CSE	34000	A

Project Operation

The Projection operation works on a single relation R and defines a relation that contains a vertical subset of R, extracting the values of specified attributes and *eliminating duplicates.*

Example – Given a relation Faculty (Class, Dept, Position) with the following tuples:

Class	Dept	Position
5	CSE	Assistant Professor
5	CSE	Assistant Professor
6	EE	Assistant Professor
6	EE	Assistant Professor

1. Project Class and Dept from Faculty –

$$\pi_{Class,\ Dept}(Faculty)$$

Class	Dept
5	CSE
6	EE

Union Operation

For R ∪ S, The union of two relations, R and S, defines a relation that contains all the tuples of R, or S, or both R and S, duplicate tuples being eliminated. R and S must be union-compatible.

For a union operation to be applied, the following rules must hold −

- r and s must have the same quantity of attributes.
- Attribute domains must be compatible.
- Duplicate tuples get automatically eliminated.

Table Employee

EMP_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21
4	SURESH	DELHI	9156768971	18

Table Student

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18

STUDENT U EMPLOYEE

ROLL_NO	NAME	ADDRESS	PHONE	AGE
1	RAM	DELHI	9455123451	18
2	RAMESH	GURGAON	9652431543	18
3	SUJIT	ROHTAK	9156253131	20
4	SURESH	DELHI	9156768971	18
5	NARESH	HISAR	9782918192	22
6	SWETA	RANCHI	9852617621	21

Set Difference

For R − S The Set difference operation defines a relation consisting of the tuples that are in relation R, but not in S. R and S must be union-compatible.

Consider the below tables Employee and Student. Perform Set Difference operation as (Employee-Student).

Employee	
Employee Id	**Employee Name**
1	Manish
2	Rohit
3	Shubhum
4	Manish

Student	
Student Id	**Student Name**
6	Sudesh
5	Deepak
2	Rishav
4	Manish
9	Aman

Step 1: Check the condition for the set difference

- Both the relation have the same number of attributes
- Data Types of Corresponding Attributes are the same

Step 2: Result after performing Set difference (Employee-Student)

Employee-Student	
Employee Id	Employee Name
1	Manish
2	Rohit
3	Shubhum

Cartesian product

For R × S, the Cartesian product operation defines a relation that is the concatenation of every tuple of relation R with every tuple of relation S.

Example: Consider two relations STUDENT(SNO, FNAME, LNAME) and DETAIL(ROLLNO, AGE) below:

SNO	FNAME	LNAME
1	Albert	Singh
2	Nora	Fatehi

ROLLNO	AGE
5	18
9	21

On applying CROSS PRODUCT on STUDENT and DETAIL:

`STUDENT × DETAILS`

SNO	FNAME	LNAME	ROLLNO	AGE
1	Albert	Singh	5	18
1	Albert	Singh	9	21
2	Nora	Fatehi	5	18
2	Nora	Fatehi	9	21

We can observe that the number of tuples in STUDENT relation is 2, and the number of tuples in DETAIL is 2. So the number of tuples in the resulting relation on performing CROSS PRODUCT is 2*2 = 4. Important points on CARTESIAN PRODUCT(CROSS PRODUCT) Operation:

1. The cardinality (number of tuples) of resulting relation from a Cross Product operation is equal to the number of attributes(say m) in the first relation multiplied by the number of attributes in the second relation(say n).

   `Cardinality = m*n`
2. The Cross Product of two relation A(R1, R2, R3, ..., Rp) with degree p, and B(S1, S2, S3, ..., Sn) with degree n, is a relation C(R1, R2, R3, ..., Rp, S1, S2, S3, ..., Sn) with degree p + n attributes.

   `Degree = p+n`
3. In SQL, CARTESIAN PRODUCT(CROSS PRODUCT) can be applied using CROSS JOIN.
4. In general, we don't use cartesian Product unnecessarily, which means without proper meaning we don't use Cartesian Product. Generally, we use Cartesian Product followed by a Selection operation and comparison on the operators as shown below :

   $\sigma_{A=D}(A \times B)$

   The above query gives meaningful results. And this combination of Select and Cross Product operation is so popular that JOIN operation is inspired by this combination.

Join Operations

Typically, you want only combinations of the Cartesian product which satisfy certain situations, and so you can normally use a Join operation instead of the Cartesian product operation. The Join operation, which combines two relations to form a new relation, is one of the essential operations in relational algebra. There are various types of Join operation, each with subtle differences, some more useful than others:

- Theta join
- Equijoin (a particular type of Theta join)
- Natural join(Natural join is just like equi-join. Only difference is that the common attribute is not included in the result twice in natural join unlike equi-join).
- Outer join
- Semijoin(Show only the rows from the left table where there is a match in the right table.)

Rename Operation

The results of relational algebra are also relations but without any name. The rename operation provides database designers to rename the output relation. The rename-operation is denoted using a small Greek letter rho (ρ).

ρ x (R)

where the symbol ‘ρ’ is used to denote the RENAME operator and R is the result of the sequence of operation or expression which is saved with the name X.

Relational Algebra Exercises

Solve all queries below using only select, project, Cartesian product, and natural join. Do not use theta-join, set operations, renaming or assignment.

Schema

Suppliers(sID, sName, address)
Parts(pID, pName, colour)
Catalog(sID, pID, price)
Catalog[sID] ⊆ Suppliers[sID]
Catalog[pID] ⊆ Parts[pID]

1. Find the names of all red parts.

$\Pi_{pName}(\sigma_{colour=\text{“red”}}Parts)$

2. Find all prices for parts that are red or green. (A part may have different prices from different manufacturers.)

$$\Pi_{price}((\sigma_{colour=\text{“}red\text{”}\vee colour=\text{“}green\text{”}} Parts) \bowtie Catalog)$$

3.Find the sIDs of all suppliers who supply a part that is red or green.

$$\Pi_{sID}((\sigma_{colour=\text{“}red\text{”}\vee colour=\text{“}green\text{”}} Parts) \bowtie Catalog)$$

4. Find the names of all suppliers who supply a part that is red or green.

$$\Pi_{sName}((\Pi_{sID}((\sigma_{colour=\text{“}red\text{”}\vee colour=\text{“}green\text{”}} Parts) \bowtie Catalog)) \bowtie Suppliers)$$