Relational Algebra

What is Relational Algebra in DBMS?

Relational algebra is a **procedural** query language that works on relational model. The purpose of a query language is to retrieve data from database or perform various operations such as insert, update, delete on the data. When I say that relational algebra is a procedural query language, it means that it tells what data to be retrieved and how to be retrieved.

On the other hand relational calculus is a non-procedural query language, which means it tells what data to be retrieved but doesn't tell how to retrieve it. We will discuss relational calculus in a separate tutorial.

Types of operations in relational algebra

We have divided these operations in two categories:
1. Basic Operations
2. Derived Operations

Basic/Fundamental Operations:

1. Select (σ)
2. Project (∏)
3. Union (∪)
4. Set Difference (-)
5. Cartesian product (X)
6. Rename (ρ)

Derived Operations:

1. Natural Join (⋈)
2. Left, Right, Full outer join (⟕, ⟗, ⟖)
3. Intersection (∩)
4. Division (÷)

Lets discuss these operations one by one with the help of examples.

Select Operator (σ)

Select Operator is denoted by sigma (σ) and it is used to find the tuples (or rows) in a relation (or table) which satisfy the given condition.

If you understand little bit of SQL then you can think of it as a where clause in SQL, which is used for the same purpose.

Syntax of Select Operator (σ)

```
σ Condition/Predicate(Relation/Table name)
```

Select Operator (σ) Example

```
Table: CUSTOMER
---------------

Customer_Id     Customer_Name      Customer_City
-----------     -------------      -------------
C10100           Steve              Agra
C10111           Raghu              Agra
C10115           Chaitanya          Noida
C10117           Ajeet              Delhi
C10118           Carl               Delhi
```

Query:

```
σ Customer_City="Agra" (CUSTOMER)
```

Output:

```
Customer_Id   Customer_Name    Customer_City
-----------   -------------    -------------
C10100        Steve            Agra
C10111        Raghu            Agra
```

Project Operator (∏)

Project operator is denoted by ∏ symbol and it is used to select desired columns (or attributes) from a table (or relation).

Project operator in relational algebra is similar to the Select statement in SQL.

Syntax of Project Operator (∏)

```
∏ column_name1, column_name2, ...., column_nameN(table_name)
```

Project Operator (∏) Example

In this example, we have a table CUSTOMER with three columns, we want to fetch only two columns of the table, which we can do with the help of Project Operator ∏.

```
Table: CUSTOMER
```

```
Customer_Id     Customer_Name      Customer_City
-----------     -------------      -------------
C10100           Steve              Agra
C10111           Raghu              Agra
C10115           Chaitanya          Noida
C10117           Ajeet              Delhi
C10118           Carl               Delhi
```

Query:

```
∏ Customer_Name, Customer_City (CUSTOMER)
```

Output:

```
Customer_Name      Customer_City
-------------      -------------
Steve              Agra
Raghu              Agra
Chaitanya          Noida
Ajeet              Delhi
Carl               Delhi
```

Union Operator (∪)

Union operator is denoted by ∪ symbol and it is used to select all the rows (tuples) from two tables (relations).

Lets discuss union operator a bit more. Lets say we have two relations R1 and R2 both have same columns and we want to select all the tuples(rows) from these relations then we can apply the union operator on these relations.

Note: The rows (tuples) that are present in both the tables will only appear once in the union set. In short you can say that there are no duplicates present after the union operation.

Syntax of Union Operator (∪)

```
table_name1 ∪ table_name2
```

Union Operator (∪) Example

Table 1: COURSE

```
Course_Id  Student_Name   Student_Id
---------  ------------   ----------
C101        Aditya         S901
C104        Aditya         S901
C106        Steve          S911
C109        Paul           S921
C115        Lucy           S931
```

Table 2: STUDENT

```
Student_Id     Student_Name   Student_Age
------------   ----------     -----------
S901           Aditya         19
S911           Steve          18
S921           Paul           19
S931           Lucy           17
S941           Carl           16
S951           Rick           18
```

Query:

```
∏ Student_Name (COURSE) ∪ ∏ Student_Name (STUDENT)
```

Output:

```
Student_Name
------------
Aditya
Carl
Paul
Lucy
Rick
Steve
```

Note: As you can see there are no duplicate names present in the output even though we had few common names in both the tables, also in the COURSE table we had the duplicate name itself.

Intersection Operator (∩)

Intersection operator is denoted by ∩ symbol and it is used to select common rows (tuples) from two tables (relations).

Lets say we have two relations R1 and R2 both have same columns and we want to select all those tuples(rows) that are present in both the relations, then in that case we can apply intersection operation on these two relations R1 ∩ R2.

Note: Only those rows that are present in both the tables will appear in the result set.

Syntax of Intersection Operator (∩)

```
table_name1 ∩ table_name2
```

Intersection Operator (∩) Example

Lets take the same example that we have taken above.
Table 1: COURSE

```
Course_Id  Student_Name   Student_Id
---------  ------------   ----------
C101        Aditya         S901
C104        Aditya         S901
C106        Steve          S911
C109        Paul           S921
C115        Lucy           S931
```

Table 2: STUDENT

```
Student_Id     Student_Name   Student_Age
------------   ----------     -----------
S901           Aditya         19
S911           Steve          18
S921           Paul           19
S931           Lucy           17
S941           Carl           16
S951           Rick           18
```

Query:

```
∏ Student_Name (COURSE) ∩ ∏ Student_Name (STUDENT)
```

Output:

```
Student_Name
------------
Aditya
Steve
Paul
Lucy
```

Set Difference (-)

Set Difference is denoted by – symbol. Lets say we have two relations R1 and R2 and we want to select all those tuples(rows) that are present in Relation R1 but **not** present in Relation R2, this can be done using Set difference R1 – R2.

Syntax of Set Difference (-)

```
table_name1 - table_name2
```

Set Difference (-) Example

Lets take the same tables COURSE and STUDENT that we have seen above.

Query:
Lets write a query to select those student names that are present in STUDENT table but not present in COURSE table.

```
∏ Student_Name (STUDENT) - ∏ Student_Name (COURSE)
```

Output:

```
Student_Name
------------
Carl
Rick
```

Cartesian product (X)

Cartesian Product is denoted by X symbol. Lets say we have two relations R1 and R2 then the cartesian product of these two relations (R1 X R2) would combine each tuple of first relation R1 with the each tuple of second relation R2. I know it sounds confusing but once we take an example of this, you will be able to understand this.

Syntax of Cartesian product (X)

```
R1 X R2
```

Cartesian product (X) Example

Table 1: R

```
Col_A    Col_B
-----    ------
AA        100
BB        200
CC        300
```

Table 2: S

```
Col_X     Col_Y
-----     -----
XX         99
YY         11
ZZ         101
```

Query:
Lets find the cartesian product of table R and S.

```
R X S
```

Output:

```
Col_A    Col_B     Col_X     Col_Y
-----    ------    ------    ------
AA        100      XX         99
AA        100      YY         11
AA        100      ZZ         101
```

```
BB        200       XX          99
BB        200       YY          11
BB        200       ZZ          101
CC        300       XX          99
CC        300       YY          11
CC        300       ZZ          101
```

Note: The number of rows in the output will always be the cross product of number of rows in each table. In our example table 1 has 3 rows and table 2 has 3 rows so the output has 3×3 = 9 rows.

Rename (ρ)

Rename (ρ) operation can be used to rename a relation or an attribute of a relation.
Rename (ρ) Syntax:
ρ(new_relation_name, old_relation_name)

Rename (ρ) Example

Lets say we have a table customer, we are fetching customer names and we are renaming the resulted relation to CUST_NAMES.

Table: CUSTOMER

```
Customer_Id     Customer_Name      Customer_City
-----------     -------------      -------------
C10100           Steve              Agra
C10111           Raghu              Agra
C10115           Chaitanya          Noida
C10117           Ajeet              Delhi
C10118           Carl               Delhi
```

Query:

```
ρ(CUST_NAMES, ∏(Customer_Name)(CUSTOMER))
```

Output:

```
CUST_NAMES
----------
Steve
Raghu
Chaitanya
Ajeet
Carl
```