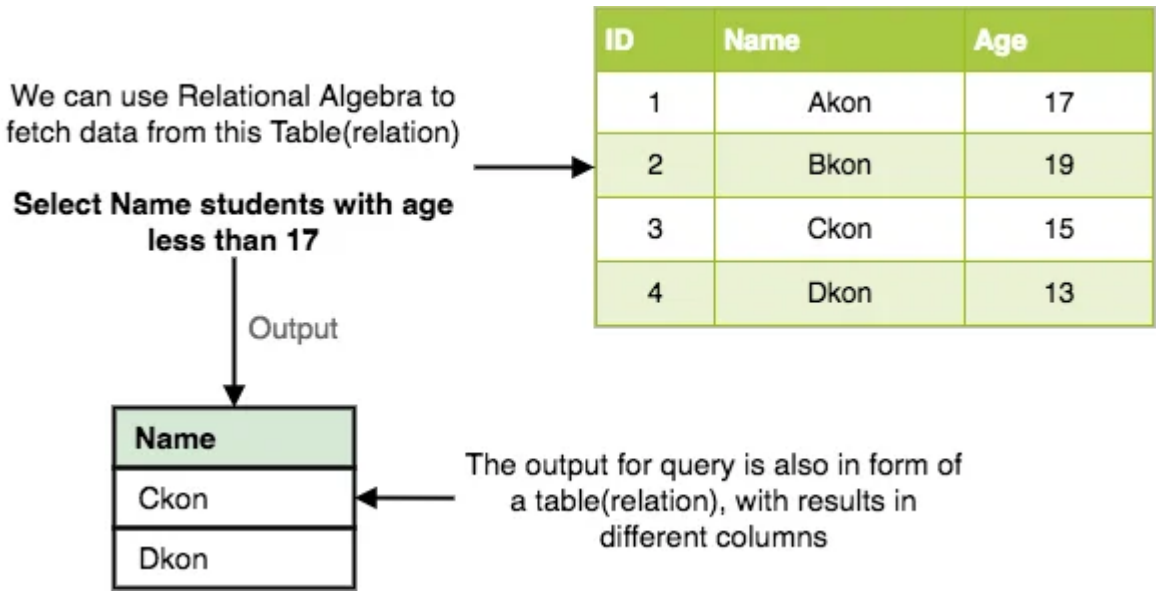
What is Relational Algebra?

Every database management system must define a query language to allow users to access the data stored in the database. **Relational Algebra** is a procedural query language used to query the database tables to access data in different

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



Relational Algebra works on the whole table at once, so we do not have to use loops etc to iterate over all the rows(tuples) of data one by one. All we have to do is specify the table name from which we need the data, and in a single line of command, relational algebra will traver Index entire given table to fetch data for you.

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

1. Select

3. Union
4. Set Different
5. Cartesian product
6. Rename

Select Operation (σ)

This is used to fetch rows(tuples) from table(relation) which satisfies a given condition.

Syntax: $\sigma_p(r)$

Where, σ represents the Select Predicate, r is the name of relation(table name in which you look for data), and p is the prepositional logic, where we specify the conditions that must be satisfied by the data. In prepositional logic, one can use **unary** and **binary** operators like =, <, >

Index

Let's take an example of the Student table we specified above in the Introduction of relational algebra, and fetch data for **students** with **age** more than 17.

$\sigma_{age > 17}$ `(Student)`

This will fetch the tuples(rows) from table **Student**, for which **age** will be greater than **17**.

You can also use, `and`, `or` etc operators, to specify two conditions, for example,

$\sigma_{age > 17 \text{ and } gender = 'Male'}$ `(Student)`

This will return tuples(rows) from table **Student** with information of male students, of age more than 17.(Consider the Student table has an attribute `Gender` too.)

Project operation is used to project only a certain set of attributes of a relation. In simple words, If you want to see only the **names** all of the students in the **Student** table, then you can use Project Operation.

It will only project or show the columns or attributes asked for, and will also remove duplicate data from the columns.

Syntax: $\Pi_{A1, A2...}(r)$

where A1, A2 etc are attribute names(column names).

Index

For example,

$\Pi_{Name, Age}(Student)$

Above statement will show us only the **Name** and **Age** columns for all the rows of data in **Student**

Union Operation (∪)

This operation is used to fetch data from two relations(tables) or temporary relation(result of another operation).

For this operation to work, the relations(tables) specified should have same number of attributes(columns) and same attribute domain.

Syntax: `A ∪ B`

where A and B are relations.

Index

For example, if we have two tables **RegularClass** and **ExtraClass**, both have a column **student** to save name of student, then,

$$\Pi_{Student}(RegularClass) \cup \Pi_{Student}(ExtraClass)$$

Above operation will give us name of **Students** who are attending both regular classes and extra classes, eliminating repetition.

Set Difference (-)

This operation is used to find data present in one relation and not present in the second relation. This operation is also applicable on two relations, just like Union operation.

Syntax: `A - B`

where A and B are relations.

Index

For example, if we want to find name of st... who attend the regular class but not the extra class, then, we can use the below operation:

$\prod_{Student}(RegularClass) - \prod_{Student}(ExtraClass)$

This is used to combine data from two different relations(tables) into one and fetch data from the combined relation.

Syntax: `A X B`

For example, if we want to find the information for Regular Class and Extra Class which are conducted during morning, then, we can use the following operation:

$\sigma_{time = 'morning'}$ (RegularClass X ExtraClass)

For the above query to work, both **RegularClass** and **ExtraClass** should have the attribute Index

Rename Operation (ρ)

This operation is used to rename the output relation for any query operation which returns result like Select, Project etc. Or to simply rename

Syntax: `ρ(RelationNew, RelationOld)`

Apart from these common operations Relational Algebra is also used for **Join** operations like,

- Natural Join
- Outer Join
- Theta join etc.

← Prev

Next →

Index

Home Code Library Test Account