Relational Algebra

Relational algebra is a procedural query language, which takes instances of relations as input and yields instances of relations as output. It uses operators to perform queries. An operator can be either **unary** or **binary**. They accept relations as their input and yield relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

The fundamental operations of relational algebra are as follows −

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

We will discuss all these operations in the following sections.

Select Operation (σ)

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

**Notation** − σ*p*(r)

Where **σ** stands for selection predicate and **r** stands for relation. *p*is prepositional logic formula which may use connectors like **and, or,** and **not**. These terms may use relational operators like − =, ≠, ≥, < ,  >,  ≤.

**For example** −

σ*subject = "database"*(Books)

**Output** − Selects tuples from books where subject is 'database'.

σsubject = "database" and price = "450"(Books)

**Output** − Selects tuples from books where subject is 'database' and 'price' is 450.

σsubject = "database" and price = "450" or year > "2010"(Books)

**Output** − Selects tuples from books where subject is 'database' and 'price' is 450 or those books published after 2010.

Project Operation (∏)

It projects column(s) that satisfy a given predicate.

Notation − ∏A1, A2, An (r)

Where A1, A2 , An are attribute names of relation **r**.

Duplicate rows are automatically eliminated, as relation is a set.

**For example** −

∏subject, author (Books)

Selects and projects columns named as subject and author from the relation Books.

Union Operation (∪)

It performs binary union between two given relations and is defined as −

r ∪ s = { t | t ∈ r or t ∈ s}

**Notation** − r U s

Where **r** and **s** are either database relations or relation result set (temporary relation).

For a union operation to be valid, the following conditions must hold −

* **r**, and **s** must have the same number of attributes.
* Attribute domains must be compatible.
* Duplicate tuples are automatically eliminated.

∏ author (Books) ∪ ∏ author (Articles)

**Output** − Projects the names of the authors who have either written a book or an article or both.

Set Difference (−)

The result of set difference operation is tuples, which are present in one relation but are not in the second relation.

**Notation** − **r** − **s**

Finds all the tuples that are present in **r** but not in **s**.

∏ author (Books) − ∏ author (Articles)

**Output** − Provides the name of authors who have written books but not articles.

Cartesian Product (Χ)

Combines information of two different relations into one.

**Notation** − r Χ s

Where **r** and **s** are relations and their output will be defined as −

r Χ s = { q t | q ∈ r and t ∈ s}

σauthor = 'tutorialspoint'(Books Χ Articles)

**Output** − Yields a relation, which shows all the books and articles written by tutorialspoint.

Rename Operation (ρ)

The results of relational algebra are also relations but without any name. The rename operation allows us to rename the output relation. 'rename' operation is denoted with small Greek letter **rho***ρ*.

**Notation** − *ρ* x (E)

Where the result of expression **E** is saved with name of **x**.

Additional operations are −

* Set intersection
* Assignment
* Natural join

Relational Calculus

In contrast to Relational Algebra, Relational Calculus is a non-procedural query language, that is, it tells what to do but never explains how to do it.

Relational calculus exists in two forms −

Tuple Relational Calculus (TRC)

Filtering variable ranges over tuples

**Notation** − {T | Condition}

Returns all tuples T that satisfies a condition.

**For example** −

{ T.name | Author(T) AND T.article = 'database' }

**Output** − Returns tuples with 'name' from Author who has written article on 'database'.

TRC can be quantified. We can use Existential (∃) and Universal Quantifiers (∀).

**For example** −

{ R| ∃T   ∈ Authors(T.article='database' AND R.name=T.name)}

**Output** − The above query will yield the same result as the previous one.

Domain Relational Calculus (DRC)

In DRC, the filtering variable uses the domain of attributes instead of entire tuple values (as done in TRC, mentioned above).

**Notation** −

{ a1, a2, a3, ..., an | P (a1, a2, a3, ... ,an)}

Where a1, a2 are attributes and **P** stands for formulae built by inner attributes.

**For example** −

{< article, page, subject > | ∈ TutorialsPoint ∧ subject = 'database'}

**Output** − Yields Article, Page, and Subject from the relation TutorialsPoint, where subject is database.

Just like TRC, DRC can also be written using existential and universal quantifiers. DRC also involves relational operators.

The expression power of Tuple Relation Calculus and Domain Relation Calculus is equivalent to Relational Algebra.

**SQL overview**

SQL is a programming language for Relational Databases. It is designed over relational algebra and tuple relational calculus. SQL comes as a package with all major distributions of RDBMS.

SQL comprises both data definition and data manipulation languages. Using the data definition properties of SQL, one can design and modify database schema, whereas data manipulation properties allows SQL to store and retrieve data from database.

Data Definition Language

SQL uses the following set of commands to define database schema −

CREATE

Creates new databases, tables and views from RDBMS.

**For example** −

Create database tutorialspoint;

Create table article;

Create view for\_students;

DROP

Drops commands, views, tables, and databases from RDBMS.

**For example**−

Drop object\_type object\_name;

Drop database tutorialspoint;

Drop table article;

Drop view for\_students;

ALTER

Modifies database schema.

Alter object\_type object\_name parameters;

**For example**−

Alter table article add subject varchar;

This command adds an attribute in the relation **article** with the name **subject** of string type.

Data Manipulation Language

SQL is equipped with data manipulation language (DML). DML modifies the database instance by inserting, updating and deleting its data. DML is responsible for all forms data modification in a database. SQL contains the following set of commands in its DML section −

* SELECT/FROM/WHERE
* INSERT INTO/VALUES
* UPDATE/SET/WHERE
* DELETE FROM/WHERE

These basic constructs allow database programmers and users to enter data and information into the database and retrieve efficiently using a number of filter options.

SELECT/FROM/WHERE

* **SELECT** − This is one of the fundamental query command of SQL. It is similar to the projection operation of relational algebra. It selects the attributes based on the condition described by WHERE clause.
* **FROM** − This clause takes a relation name as an argument from which attributes are to be selected/projected. In case more than one relation names are given, this clause corresponds to Cartesian product.
* **WHERE** − This clause defines predicate or conditions, which must match in order to qualify the attributes to be projected.

**For example** −

Select author\_name

From book\_author

Where age > 50;

This command will yield the names of authors from the relation**book\_author** whose age is greater than 50.

INSERT INTO/VALUES

This command is used for inserting values into the rows of a table (relation).

**Syntax**−

INSERT INTO table (column1 [, column2, column3 ... ]) VALUES (value1 [, value2, value3 ... ])

Or

INSERT INTO table VALUES (value1, [value2, ... ])

**For example** −

INSERT INTO tutorialspoint (Author, Subject) VALUES ("anonymous", "computers");

UPDATE/SET/WHERE

This command is used for updating or modifying the values of columns in a table (relation).

**Syntax** −

UPDATE table\_name SET column\_name = value [, column\_name = value ...] [WHERE condition]

**For example** −

UPDATE tutorialspoint SET Author="webmaster" WHERE Author="anonymous";

DELETE/FROM/WHERE

This command is used for removing one or more rows from a table (relation).

**Syntax** −

DELETE FROM table\_name [WHERE condition];

**For example** −

DELETE FROM tutorialspoints

WHERE Author="unknown";