Relational database systems are expected to be equipped with a query language that can assist its users to query the database instances. There are two kinds of query languages − relational algebra and relational calculus.

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 ( )

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

Notation − p(r)

Where 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 −

```
subject = "database"(Books)
```

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

```
subject = "database" and price = "450"(Books)
```

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

```
subject = "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 (X)

Combines information of two different relations into one.

Notation − r X s

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

$rXs = qt|q \in randt \in s$

```
~author~ = 'tutorialspoint'(Books X Articles)
```

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

Rename Operation ( )

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** .

Notation − 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 −

$T.name|Author(T)ANDT.article =' database'$

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 −

$R|\exists T \in Authors(T.article =' database' AND R.name = T.name)$

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 −

$< article, page, subject > | \in TutorialsPoint \wedge subject =' database'$

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.