Relational Algebra

Relational database systems are expected to be equipped by a query language that can assist its user to query the database instances. This way its user empowers itself and can populate the results as required. 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 yields relations as their output. Relational algebra is performed recursively on a relation and intermediate results are also considered relations.

Fundamental operations of Relational algebra:

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

These are defined briefly as follows:

Select Operation (σ)

Selects tuples that satisfy the given predicate from a relation.

Notation $\sigma_p(r)$
Where *p* stands for selection predicate and r stands for relation. *p* is prepositional logic formulae which may use connectors like and, or and not. These terms may use relational operators like: =, ≠, ≥, < , >, ≤.

For example:

```
σsubject="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 the publication year is greater than 2010, that is published after 2010.

Project Operation (∏)

Projects column(s) that satisfy given predicate.

Notation: $\prod_{A_1, A_2, A_n}$ (r)
Where a_1, a_2 , a_n 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 relation Books.

Union Operation (∪)

Union operation performs binary union between two given relations and is defined as:

```
r ∪ s = { t | t ∈ r or t ∈ s}
```

Notion: 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, s must have same number of attributes.
- Attribute domains must be compatible.

Duplicate tuples are automatically eliminated.

```
∏ author (Books) ∪ ∏ author (Articles)
```

Output : Projects the name of author who has either written a book or an article or both.

Set Difference (−)

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

Notation: r − s

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

```
∏ author (Books) − ∏ author (Articles)
```

Output: Results the name of authors who has 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 there output will be defined as:

r X s = { q t | q ∈ r and t ∈ s}

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

Output : yields a relation as result which shows all books and articles written by tutorialspoint.

Rename operation (ρ)

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 with Relational Algebra, Relational Calculus is non-procedural query language, that is, it tells what to do but never explains the way, 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 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 also. We can use Existential (∃)and Universal Quantifiers (∀).

For example:

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

Output : the query will yield the same result as the previous one.

Domain relational calculus (DRC)

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

Notation:

{ a_1, a_2, a_3, ..., a_n | P (a_1, a_2, a_3, ... ,a_n)}

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 relation TutorialsPoint where Subject is database.

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

Expression power of Tuple relation calculus and Domain relation calculus is equivalent to Relational Algebra.

Source:

http://www.tutorialspoint.com/dbms/relational_algebra.htm