Relational Calculus

There is an alternate way of formulating queries known as Relational Calculus. Relational calculus is a non-procedural query language. In the non-procedural query language, the user is concerned with the details of how to obtain the end results. The relational calculus tells what to do but never explains how to do. Most commercial relational languages are based on aspects of relational calculus including SQL-QBE and QUEL.

Why it is called Relational Calculus?

It is based on Predicate calculus, a name derived from branch of symbolic language. A predicate is a truth-valued function with arguments. On substituting values for the arguments, the function result in an expression called a proposition. It can be either true or false. It is a tailored version of a subset of the Predicate Calculus to communicate with the relational database.

**Many of the calculus expressions involves the use of Quantifiers. There are two types of quantifiers:**

* **Universal Quantifiers:** The universal quantifier denoted by ∀ is read as for all which means that in a given set of tuples exactly all tuples satisfy a given condition.
* **Existential Quantifiers:** The existential quantifier denoted by ∃ is read as for all which means that in a given set of tuples there is at least one occurrences whose value satisfy a given condition.

Before using the concept of quantifiers in formulas, we need to know the concept of Free and Bound Variables.

A tuple variable t is bound if it is quantified which means that if it appears in any occurrences a variable that is not bound is said to be free.

Free and bound variables may be compared with global and local variable of programming languages

.**Types of Relational calculus:**

### DBMS Relational Calculus

### Tuple Relational Calculus (TRC)

It is a non-procedural query language which is based on finding a number of tuple variables also known as range variable for which predicate holds true. It describes the desired information without giving a specific procedure for obtaining that information. The tuple relational calculus is specified to select the tuples in a relation. In TRC, filtering variable uses the tuples of a relation. The result of the relation can have one or more tuples.

**Notation:**

A Query in the tuple relational calculus is expressed as following notation

1. {T | P (T)}   or {T | Condition (T)}

Where

**T** is the resulting tuples

**P(T)** is the condition used to fetch T.

**For example:**

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

**Output:** This query selects the tuples from the AUTHOR relation. It returns a tuple with 'name' from Author who has written an article on 'database'.

TRC (tuple relation calculus) can be quantified. In TRC, we can use Existential (∃) and Universal Quantifiers (∀).

**For example:**

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

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

2. Domain Relational Calculus (DRC)

The second form of relation is known as Domain relational calculus. In domain relational calculus, filtering variable uses the domain of attributes. Domain relational calculus uses the same operators as tuple calculus. It uses logical connectives ∧ (and), ∨ (or) and ┓ (not). It uses Existential (∃) and Universal Quantifiers (∀) to bind the variable. The QBE or Query by example is a query language related to domain relational calculus.

**Notation:**

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

Where

**a1, a2** are attributes  
**P** stands for formula built by inner attributes

**For example:**

1. {< article, page, subject > |  ∈ javatpoint ∧ subject = 'database'}

**Output:** This query will yield the article, page, and subject from the relational javatpoint, where the subject is a database.