Relational Calculus in Detail

Relational calculus is a **non-procedural (declarative) query language** in DBMS that allows users to specify what data they want from the database without describing how to retrieve it. It is grounded in predicate logic (first-order logic) and forms a theoretical foundation for SQL and other high-level query languages.

Types of Relational Calculus

There are two main types:

- **Tuple Relational Calculus (TRC)**
- **Domain Relational Calculus (DRC)**

Tuple Relational Calculus (TRC)

TRC uses **tuple variables** to represent rows in a relation. The query specifies the set of all tuples for which a given predicate (logical condition) is true.

General Syntax:

$$\{t \mid P(t)\}$$

Where:

- $ t $ is a tuple variable that iterates over each row of a relation.
- $ P(t) $ is a predicate (logical expression) that must be satisfied.

Example:

Retrieve all customers with Zip code 12345 from the Customer table:

$$\{t \mid t \in Customer \land t.Zipcode = 12345\}$$

This returns all tuples $ t $ from the Customer table where the Zipcode is 12345[1][4][3].

Features:

- Focuses on tuples (rows).
- Uses logical connectives (AND, OR, NOT) and quantifiers (∃ for "exists", ∀ for "for all").
- Returns sets of tuples that satisfy the condition.

Domain Relational Calculus (DRC)

DRC uses **domain variables** that represent values from attributes (columns) of relations. The query specifies the set of attribute values for which a predicate is true.

General Syntax:

$$\{\langle x_1, x_2, \ldots, x_n \rangle \mid P(x_1, x_2, \ldots, x_n)\}$$

Where:

- $ x_1, x_2, ..., x_n $ are domain variables corresponding to attributes.
- $ P(x_1, x_2, ..., x_n) $ is a predicate.

Example:
Retrieve all customer data with Zip code 12345:

$$\{\langle x_1, x_2, x_3 \rangle \mid \langle x_1, x_2, x_3 \rangle \in Customer \wedge x_3 = 12345\}$$

This returns all attribute value combinations where the third value (Zip code) is 12345.

Features:

- Focuses on individual attribute values.
- Uses logical connectives and quantifiers.
- Returns sets of attribute value combinations.

Key Points and Comparison

- **Non-procedural:** Users specify *what* they want, not *how* to get it.
- **Based on Predicate Logic:** Both forms use logical expressions to describe the desired data.
- **Safety:** Only "safe" expressions (those that return finite results) are valid in practice.
- **Influence:** Relational calculus concepts are foundational for SQL and query-by-example systems.