Database Engineering

Lecture #6

Relational Calculus

Presented By:
Dr. Suvasini Panigrahi
Associate Professor, Department of CSE,
VSSUT, Burla

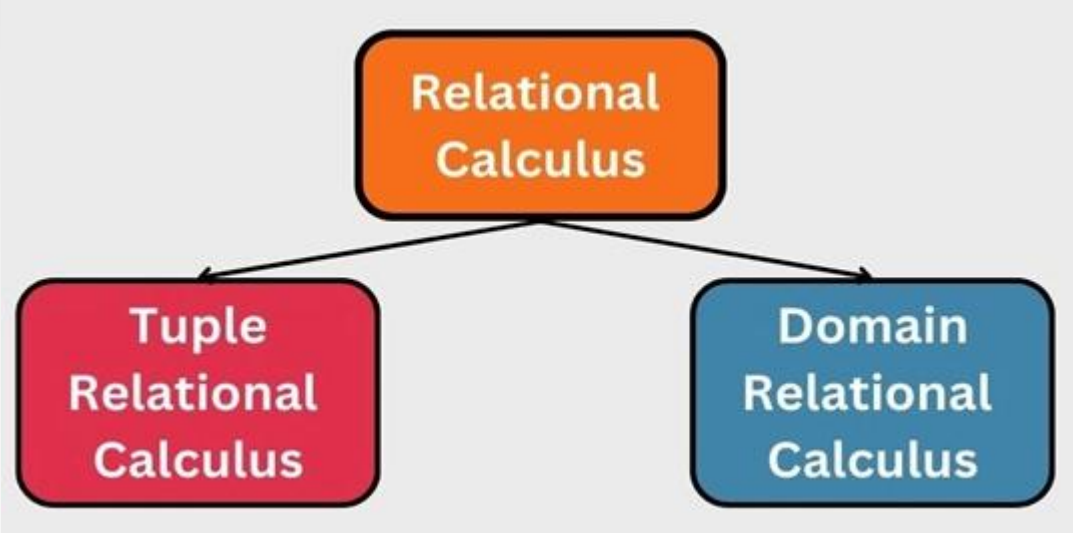
Database System Concepts, 5th Ed.

Outline

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

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, it guides the users on what data is needed from the database without specifying how to obtain it.
- **Types of Relational calculus:**



Tuple Relational Calculus

Tuple Relational Calculus

- Tuple Relational Calculus (TRC) is a non-procedural/declarative query language used in relational database management systems (RDBMS) to retrieve data from tables.
- TRC is based on the concept of tuple variables.
- As it is a non-procedural language, hence, it specifies what data is required from the database, rather than how to retrieve it.
- In TRC, each query is of the following form:

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

  where, *t* is a tuple variable and *P(t)* is a predicate/tuple calculus formula that describes the conditions that the tuples in the result must satisfy. Thus, *t* is the set of all tuples such that predicate *P* is true for *t.*
- The curly braces {} are used to indicate that the result is a set of tuples.
- *t* [*A*] denotes the value of tuple *t* on attribute *A*

Tuple Calculus Formula

1. Set of attributes and constants
2. Set of comparison operators: (e.g., $<$, $\leq$, $=$, $\neq$, $>$, $\geq$)
3. Set of connectives: and ($\wedge$), or (v), not ($\neg$)
4. Implication ($\Rightarrow$): x $\Rightarrow$ y, if x if true, then y is true

$$x \Rightarrow y \equiv \neg x \text{ v } y$$

5. Set of quantifiers:
   - $\exists\, t \in r\ (Q\ (t\ )) \equiv$ "there exists" a tuple t in relation r such that predicate $Q\ (t\ )$ is true
   - $\forall t \in r\ (Q\ (t\ )) \equiv Q$ is true "for all" tuples t in relation r

Banking Example

- *branch* (*branch_name, branch_city, assets*)
- *customer* (*customer_name, customer_street, customer_city*)
- *account* (*account_number, branch_name, balance*)
- *loan* (*loan_number, branch_name, amount*)
- *depositor* (*customer_name, account_number*)
- *borrower* (*customer_name, loan_number*)

Example Queries

- Find the *loan_number, branch_name,* and *amount* for loans of over $1200.

$$\{t \mid t \in loan \wedge t\,[amount\,] > 1200\}$$

- Find the loan number for each loan of an amount greater than $1200.

$$\{t \mid \exists\, s \in \text{loan}\ (t\,[loan\_number\,] = s\,[loan\_number\,] \wedge s\,[amount\,] > 1200)\}$$

Example Queries

- Find the names of all customers having a loan, an account, or both at the bank.

$$\{t \mid \exists s \in \textit{borrower}\ (\ t\ [\textit{customer\_name}\ ] = s\ [\textit{customer\_name}\ ])$$
$$\vee \exists u \in \textit{depositor}\ (\ t\ [\textit{customer\_name}\ ] = u\ [\textit{customer\_name}\ ])$$

- Find the names of all customers who have a loan and an account at the bank.

$$\{t \mid \exists s \in \textit{borrower}\ (\ t\ [\textit{customer\_name}\ ] = s\ [\textit{customer\_name}\ ])$$
$$\wedge \exists u \in \textit{depositor}\ (\ t\ [\textit{customer\_name}\ ] = u\ [\textit{customer\_name}]\ )$$

Example Queries

- Find the names of all customers having a loan at the Delhi branch.

$$\{t \mid \exists s \in \textit{borrower}\ (t[\textit{customer\_name}] = s[\textit{customer\_name}] \wedge \exists u \in \textit{loan}\ (u[\textit{branch\_name}] = \text{“Delhi”} \wedge u[\textit{loan\_number}] = s[\textit{loan\_number}]))\}$$

- Find the names of all customers who have an account at the bank but do not have a loan from the bank.

$$\{t \mid \exists u \in \textit{depositor}\ (t[\textit{customer\_name}] = u[\text{customer\_name}] \wedge \neg \exists s \in \textit{borrower}\ (t[\textit{customer\_name}] = s[\textit{customer\_name}])\}$$

Safety of Expressions

- It is possible to write tuple calculus expressions that generate infinite relations.
- For example, { t | $\neg$ ($t \in r$) } results in an infinite relation as there are infinite number of tuples that are not in *r*.
- To guard against the problem, we restrict the set of allowable expressions to safe expressions.
- To help us define a restriction of the tuple relational calculus, we introduce the concept of the domain of a tuple relational formula, *P.*
- *The domain of P,* denoted *as dom(P), is the set of all values referenced by P.*
- An expression {t | P (t)} in the tuple relational calculus is *safe* if it takes values from the domain of the tuple calculus formula P to produce a finite number of tuples as its result. Otherwise, it is called unsafe.

Domain Relational Calculus

- The second form of relation is known as Domain Relational Calculus (DRC).
- DRC is a non-procedural query language equivalent in power to the Tuple Relational Calculus.
- DRC uses the list of attributes to be selected from the relation based on the condition. It is the same as TRC but differs by selecting the attributes rather than selecting whole tuples.
- In DRC, each query is an expression of the form:

$$\{ < x_1, x_2, \ldots, x_n > \mid P (x_1, x_2, \ldots, x_n)\}$$

  - $x_1, x_2, \ldots, x_n$ represent domain variables (attributes of a relation)
  - *P* represents a formula/condition similar to that of the predicate calculus
- *Domain variables* take values from an attributes domain rather than values as entire tuple(s).

Example Queries

- Find the *loan_number, branch_name,* and *amount* for loans of over $1200.

$$\{< l, b, a > \mid < l, b, a > \in loan \wedge a > 1200\}$$

- Find the names of all customers who have a loan of over $1200.

$$\{< c > \mid \exists\, l, b, a\, (< c, l > \in borrower \wedge < l, b, a > \in loan \wedge a > 1200)\}$$

- Find the names of all customers who have a loan from the Delhi branch and the loan amount.
  - $\{< c, a > \mid \exists\, l\, (< c, l > \in borrower \wedge \exists b\, (< l, b, a > \in loan \wedge b = \text{“Delhi ”}))\}$
  - $\{< c, a > \mid \exists\, l\, (< c, l > \in borrower \wedge < l, \text{“Delhi”}, a > \in loan)\}$

Example Queries

- Find the names of all customers having a loan, an account, or both at the Delhi branch:

$$\{< c > \mid \exists\, l\, (< c, l > \in \textit{borrower} \wedge \exists\, b,a\, (< l, b, a > \in \textit{loan} \wedge b = \text{“Delhi ”})) \vee \exists\, a\, (< c, a > \in \textit{depositor} \wedge \exists\, b,n\, (< a, b, n > \in \textit{account} \wedge b = \text{“Delhi ”}))\}$$