Relational Calculus

Course Instructor-

Dr. Koushlendra Kumar Singh

Assistant Professor

Department of Computer Science and Engineering

National Institute of Jamshedpur- Jharkhand

Relational calculus

- It is a nonprocedural query language.
- It describes the desired information without giving specific procedure for obtaining that information.
- There are two versions of the relational calculus:
  - **Tuple relational calculus (TRC)**
  - **Domain relational calculus (DRC)**
- Both TRC and DRC are simple subsets of first-order logic.
- The difference is the level at which variables are used: for fields (domains) or for tuples.
- The calculus is non-procedural (‘declarative’) compared to the relational algebra.

Domain Relational Calculus

- A query in tuple relational calculus is expressed as

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

- That is, it is the set of all tuples t such that predicate P is true for t.

OR

Queries have the form

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

where $x_1,\ldots,x_n$ are domain variables and F is a formula with free variables $\{x_1,\ldots,x_n\}$

Answer: all tuples $<v_1,\ldots,v_n>$ that make $F(v_1,\ldots,v_n)$ true.

- **Formula** is recursively defined:
  - start with simple atomic formulas

    (get tuples from relations or make comparisons of values)
  - build bigger formulas using logical connectives.

TRC Formulas

- An Atomic formula is one of the following:

  $R \in Rel$

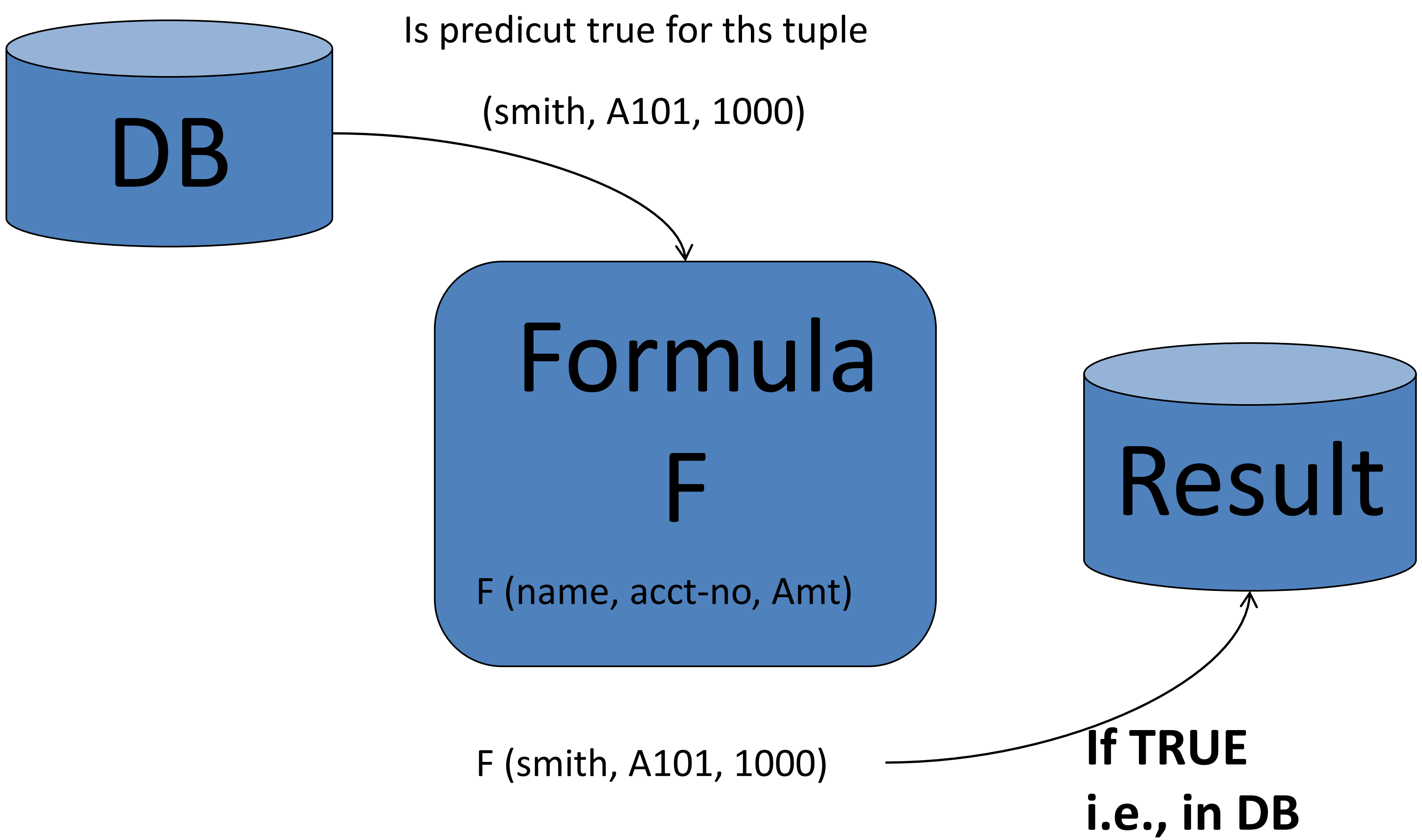
  R[a] op S[b] <u>*or*</u> *R.a =S.b*

  R[a] op constant *where R[a] denotes attr.a of Rel R.*

  Where op is one of $<,>,=,\leq,\geq,\neq$

- A formula can be:
  - an atomic formula
  - $\neg p, p \wedge q, p \vee q$ where p and q are formulas
  - $\exists R(p(R))$ where variable *R* is a tuple variable
  - $\forall R(p(R))$ where variable *R* is a tuple variable

Relational Calculus



Free and Bound Variables

- Quantifiers $\exists X$ and $\forall X$ in a formula are said to

  bind X in the formula.

  A variable that is not bound is free.
- Let us revisit the definition of a query:
  - {T | p(T)}

- **Important restriction**
  - the variable *T* that appears to the left of `|' must be the *only* free variable in the formula *p(T)*.
  - in other words, all other tuple variables must be bound using a quantifier.

Examples Schema

Sailors (sid, sname, age, rating)

Boats (bid, color)

Reserves (sid, bid)

Selection and Projection

- Find all sailors with rating above 7.

$$\{S \mid S \in Sailors \wedge S[rating] > 7\}$$

Note: Modify this query to answer: Find sailors who are older than 18 or have a rating under 9, and are named 'Bob'.

Examples

- Find names and ages of sailors with rating above 7.

$$\{S \mid \exists S1 \in Sailors(S1[rating] > 7 \wedge S[sname] = S1[sname] \wedge S[age] = S1[age])\}$$

- Note: *S* is a tuple variable with 2 attributes (i.e. {S} is a projection of *Sailors*) only 2 attributes are ever mentioned and *S* is never used to range over any relations in the query.

Examples: Joins

Find sailors and their rating for sailors rated > 7 who have reserved boat #103

$$\{S \mid S \in Sailors \wedge S[rating] > 7 \wedge \exists R \in Reserves\ (R[sid] = S[sid] \wedge R[bid] = 103)\}$$

Note the use of $\exists$ to find a tuple in Reserves that `joins with' the Sailors tuple under consideration.

Examples: Joins

Find sailors rated > 7 who've reserved a red boat

$$\{S \mid S \in Sailors \wedge S[rating] > 7 \wedge \exists R \in Reserves\ (R[sid] = S[sid] \wedge \exists B \in Boats\ (B[bid] = R[bid] \wedge B[color] = \text{'red'}))\}$$

Division

Find sailors who've reserved all boats.

$$\{S \mid S \in Sailors \wedge \forall B \in Boats\ (\exists R \in Reserves\ (S[sid] = R[sid] \wedge B[bid] = R[bid]))\}$$

Find all sailors *S* such that for all tuples *B* in Boats there is a tuple in Reserves showing that sailor *S* has reserved *B*.

Unsafe Queries, Expressive Power

- ∃ syntactically correct calculus queries that have an infinite number of answers! *Unsafe* queries.
  - e.g., $\left\{ S \mid \neg \left(S \in Sailors \right) \right\}$
  - Solution???? Don't do that!

- *Expressive Power:*
  - every query that can be expressed in relational algebra can be expressed as a *safe* query in DRC / TRC; the converse is also true.

- *Relational Completeness:* Query language (e.g., SQL) can express every query that is expressible in relational algebra/calculus. (actually, SQL is more powerful)

Tuple Relational Calculus(Join Queries)

Find the names of customers w/ loans at the Perry branch.

Answer has form $\{t \mid P(t)\}$.

Strategy for determining *P*(*t*):

1. What tables are involved?

   borrower (*s*), *loan* (*u*)

2. What are the conditions?

   (a) Projection: $t\,[cname] = s\,[cname]$
   (b) Join: $s\,[lno] = u\,[lno]$
   (c) Selection: $u\,[bname] =$ "Perry"

Tuple Relational Calculus(Join Queries)

Find the names of customers w/ loans at the Perry branch.

A. $\{t \mid \exists\, s \in borrower\ (P(t,s))\}$ such that:

$$P(t,s) \quad \equiv t\,[cname] = s\,[cname] \wedge \exists\, u \in loan\ (Q(t,s,u))$$
$$Q(t,s,u) \quad \equiv s\,[lno] = u\,[lno] \wedge\ u\,[bname] = \text{“Perry”}$$

OR *unfolded version (either is ok)*

$$\{t \mid \exists\, s \in borrower\ (\\ t\,[cname] = s\,[cname] \wedge \\ \exists\, u \in loan\ (s\,[lno] = u\,[lno] \wedge u\,[bname] = \text{“Perry”}))\}$$

Thank You ????

Email: koushlendra.cse@nitjsr.ac.in