CS589 Principles of DB Systems
Lecture 2: Relational Calculus

David Maier (maier@cs.pdx.edu)

Goals for this lecture

- Introduce tuple relational calculus queries (based on Ramakrishnan & Gehrke)
- Formal definition of tuple relational calculus
- Introduce domain relational calculus queries (based on Levene & Loizou)
- Formal definition of domain relational calculus

Example tuple calculus query

Student(s-id, name, major, f-id, age)

Differences between Ramakrishnan/Gehrke & Levene/Loizou:

1. Capital letters for tuple variables rather than lower case letters.
2. Using the relation name (e.g., Student) to represent the current relation (what is called "r" in the textbook).
3. Using the notation T.major to refer to one value in a tuple rather than t(major) as in the textbook.

$\{ T \mid T \in \text{Student} \wedge \text{T.age} > 30 \wedge \text{T.major} = \text{"CS"}\}$

What does this query find for us?

Relational algebra vs. relational calculus

A relational algebra query is a well-formed expression with relational algebra operators

Student(s-id, name, major, f-id, age)

$\sigma_{age<20}\text{Student}) \cup \sigma_{age>50}\text{Student})$

A tuple relational calculus query is a set-definition of form

$\{ T \mid F(T)\}$

where T is a tuple variable and F is a logical expression, with T as the only free variable, that evaluates to true or false.

$\{ T \mid T \in \text{Student} \wedge (\text{Age} < 20 \vee \text{Age} > 50) \}$

In SQL,

Select * From Student S Where (S.Age < 20 or S.Age > 50)

More examples of tuple calculus queries

Student(s-id, name, major, f-id, age)
Faculty(f-id, name, rank, dept)

{ T | T ∈ Faculty ∧ (rank = "Prof" ∨ rank = "Assoc Prof")}

{ T | T ∈ Student ∧ T ∈ Faculty } Not well-formed; why?

{ T | ∃F ∈ Faculty ∃S ∈ Student (F.name = S.name ∧ F.dept = S.major ∧ T.name = F.name ∧ T.dept = S.major}
What is schema for T?

{ T | ∃F ∈ Faculty ∃S ∈ Student (F.name = S.name ∧ F.dept = S.major ∧ T.name = S.name ∧ T.dept = F.dept)}
Note: this query and the preceding one are equivalent.

{ T | ∃S ∈ Student ∃F ∈ Faculty (F.f-id = S.f-id ∧ T.faculty-name = F.name ∧ T.student-name = S.name)}

More tuple calculus examples

Undergrad(id, name, phone)
Grad(id, name, phone)

{ V | V ∈ Undergrad ∧ V ∈ Grad } is what?

{ W | W ∈ Undergrad ∨ W ∈ Grad } is what?

{ T | V ∈ Undergrad ∧ T.name = V.name} is what?

{ T | T.id = V.id } is what?

In-class exercise

Explain each of these relational algebra queries in English and write each of them in tuple calculus.

Student(s-id, name, major, f-id, age)
Faculty(f-id, name, rank, dept)

1. $(\pi_{name}\sigma_{age=21}Student)) \cup \pi_{name}Faculty)$
2. $Student \bowtie Faculty$
3. $(\pi_{name}\sigma_{age=21}Student)) - (\pi_{name}\sigma_{major=\text{"CS"}}Student))$

More tuple calculus examples

Student(s-id, name, major, f-id, age)
Faculty(f-id, name, rank, dept)

have ≥ 1 advisee

$\{ T \mid T \in Faculty \wedge \exists S \in Student\ (S.f\text{-}id = T.f\text{-}id) \}$

$\{ T \mid T \in Faculty \wedge \forall S \in Student\ (S.f\text{-}id = T.f\text{-}id) \}$

advises all students

What does each query find for us?

How would you express each in relational algebra?

Definition of tuple-calculus syntax (Ramakrishnan & Gehrke)

Rel is a relation name, *R* and *S* are tuple variables, *a* is an attribute of *R*, *b* is an attribute of *S*, *op* is one of {<, >, =, ≤, ≥, ≠}

An atomic formula is one of the following:

- *R* ∈ *Rel*
- *R.a op S.b*
- *R.a op constant* (or *constant op R.a*)

A formula is:

- any atomic formula
- ¬F, (F), F1∧F2, F1∨F2, F1⇒F2 (if F, F1, and F2 are formula)
- ∃R(F(R)) where F is formula with R a free variable
- ∀R(F(R)) where F is a formula with R a free variable

A tuple calculus query is an expression of the form:
{ T | F(T) } where T is the only free variable in F

Note: all of the variables are tuple variables.

Bound and free variables

See pages 35-37 in our textbook.

Free variables in a formula are defined as:

1. All the variables occurring in an atomic formula are free.
2. The free variables in F ∧ G are the free variables of F plus the free variables of G.
3. The free variables in F ∨ G are the free variables of F plus the free variables of G.
4. The free variables in (F) or ¬F are the free variables of F.
5. In ∃xF or ∀xF, the free variables are the free variables of F except for x. We say x is a bound variable when it appears with ∃ or ∀.

Free and Bound Variable Example

{ T | T ∈ Student ∧
∃M ∈ Faculty (T.f-id = M.f-id ∧
∀S ∈ Student
(S.dept ≠ T.dept ∨ S.f-id = M.f-id))}

Shortcut: ∃M ∈ Faculty (F(M))

Definition of tuple calculus syntax (Ramakrishnan & Gehrke)

Rel is a relation name, *R* and *S* are tuple variables, *a* is an attribute of *R*, *b* is an attribute of *S*, *op* is one of {<, >, =, ≤, ≥, ≠}

An atomic formula is one of the following:

- *R* ∈ *Rel*
- *R.a op S.b*
- *R.a op constant* (or *constant op R.a*)

A formula is:

- any atomic formula
- ¬F, (F), F1∧F2, F1∨F2, F1⇒F2 (if F, F1, and F2 are formula)
- ∃R(F(R)) where F is formula with R a free variable
- ∀R(F(R)) where F is a formula with R a free variable

A tuple calculus query is an expression of the form:
{ T | F(T) } where T is the only free variable in F

Note: all of the variables are tuple variables.

Domain calculus uses domain variables

Domain calculus is very similar to tuple calculus.

We use domain variables rather than tuple variables.

Tuples can be substituted for tuple variables.

Domain values (like 6 or “Smith”) can be substituted for domain variables.

Domain Relational Calculus Expression

distinct attributes names
(in the query answer)
$A_1, A_2, \ldots, A_n$

$$\{x_1{:}A_1,\ x_2{:}A_2,\ \ldots\ ,\ x_n{:}A_n \mid F(x_1, x_2, \ldots, x_n)\ \}$$

$x_1, x_2, \ldots, x_n$
domain variables

$F(x_1, x_1, \ldots, x_n)$
an expression in
logic where x1, x1, ..., xn
are (the only) free variables

Example domain calculus queries

Student(s-id, name, major, f-id, age)
Faculty(f-id, name, rank, dept)

1. { x : s-id, y : name, z : major | Student(x, y, z, 21) }
2. { x : name, y : name | ∃i:s-id ∃m:major ∃f:f-id ∃a:age ∃r:rank ∃d:dept (Student(i, x, m, f, a) $\wedge$ Faculty (f, y, r, d)) }
3. { x : name | ∃i:s-id ∃m:major ∃f:f-id ∃a:age ∃r:rank ∃d:dept (Student(i, x, m, f, a) $\wedge$ Faculty (f, x, r, d)) }

Exercise

Write each of the following queries in domain calculus.

Student(s-id, name, major, f-id, age)
Faculty(f-id, name, rank, dept)

1. $(\pi_{name}\sigma_{age=21}$Student$)) \cup \pi_{name}$Faculty)
2. Student $\bowtie$ Faculty
3. $(\pi_{name}\sigma_{age=21}$Student$))$ – $(\pi_{name}\sigma_{major=\text{“CS”}}$Student$))$

Answers for a Domain Relational Calculus Expression

$\{x_1{:}A_1, x_2{:}A_2, \ldots, x_n{:}A_n \mid F(x_1, x_2, \ldots, x_n)\}$

The query answer for a database $d = \{r_1, r_2, \ldots, r_m\}$ over the database schema $\mathcal{R} = \{R_1, R_2, \ldots, R_m\}$ is a relation r over relation schema R with schema(R) = $\{A_1, A_2, \ldots, A_n\}$ such that a tuple $<v_1, v_2, \ldots, v_n> \in r$ iff

1. for all $i \in \{1, 2, \ldots, n\}$, $v_i \in DOM(A_i)$ and
2. if for all $i \in \{1, 2, \ldots, n\}$, we substitute v_i for x_i in F, then $<v_1, v_2, \ldots, v_n>$ satisfies F with respect to the database d.

Details

$\{x_1{:}A_1, x_2{:}A_2 \mid F(x_1, x_2)\}(d)$

- In our textbook, a domain calculus query is followed by the symbol "d" which is the database over which the query is being evaluated. Sometimes the query is followed by a set of relations, e.g., ({s, r})
- The definition of formulas and bound and free variables is essentially the same for domain calculus (as described in our textbook) and tuple calculus (as described in Ramakrishnan & Gehrke). I expect you to be able to read and understand this material.

Satisfaction of a formula by a tuple

Let d = $\{r_1, r_2, ..., r_m\}$ be a database over the database schema $\mathcal{R}$ = $\{R_1, R_2, ..., R_m\}$. Given a query $\{x_1{:}A_1, x_2{:}A_2, ... , x_n{:}A_n \mid F(x_1, x_2, ..., x_n) \}$

a tuple $<v_1, v_2, ..., v_n>$ satisfies the formula F with respect to d, if for all i $\in \{1, 2, ..., n\}$, $v_i \in DOM(A_i)$ and one of the following is satisfied:

1. If F is the atomic formula $R(y_1, y_2, ..., y_k)$ then $R \in \mathcal{R}$ and the tuple t with v_i substituted for each variable y_i satisfies $t \in r$ where r is the relation over R in d.
2. If F is the atomic formula $x_i = y_j$, then $v_i = v_j$ is satisfied where v_i is substituted for x_i and either y_j is a variable and v_j is substituted for it or y_j is a constant and $v_j = y_j$.
3. If F is the formula (G), then $<v_1, v_2, ..., v_n>$ satisfies the formula F if $<v_1, v_2, ..., v_n>$ satisfies the formula G.

Satisfaction (cont.)

4. If F has the form $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, or $F_1 \Rightarrow F_2$, then $<v_1, v_2, ..., v_n>$ satisifes F according to the logical connectors.

5. If F is the formula $\exists x_i{:}A\ (G(x_1, x_2, ..., x_i, ..., x_n)$, then $<v_1, v_2, ..., v_{i-1}, v_{i+1}, ...v_n>$ satisifes F if there exists a constant $v_i \in DOM(A)$ such than when v_i is substituted for x_i, $<v_1, v_2, ..., v_{i-1}, v_i, v_{i+1}, ... v_n>$ satisfies G.

6. If F is the formula $\forall x_i{:}A\ (G(x_1, x_2, ..., x_i, ..., x_n)$, then $<v_1, v_2, ..., v_{i-1}, v_{i+1}, ...v_n>$ satisifes F if for all constants $v_i \in DOM(A)$, when v_i is subsituted for x_i, $<v_1, v_2, ..., v_n>$ satisifes G.

Redo: Satisfaction of a formula by a tuple

Let d = $\{r_1, r_2, ..., r_m\}$ be a database over the database schema $\mathcal{R}$ = $\{R_1, R_2, ..., R_m\}$. Given a query
$\{x_1{:}A_1, x_2{:}A_2, ... , x_n{:}A_n \mid F(x_1, x_2, ..., x_n) \}$

a tuple v = $<v_1, v_2, ..., v_n>$ satisfies the formula F with respect to d, if for all i $\in$ {1, 2, ..., n}, $v_i \in$ DOM(A_i) and one of the following is satisfied:

1. If F is the atomic formula $R_j(x_{i1}, x_{i2}, ..., x_{ik})$ then $<v_{i1}, v_{i2}, ..., v_{ik}> \in r_j$.
2. i. If F is the atomic formula $x_i = x_j$, then $v_i = v_j$
   ii. If F is the atomic formula $x_i = a$, then $v_i = a$.
3. If F is the formula (G), then v satisfies the formula F if v satisfies the formula G.

Redo: Satisfaction (cont.)

4. If F has the form $\neg F$, $F_1 \wedge F_2$, $F_1 \vee F_2$, or $F_1 \Rightarrow F_2$, then v satisifes F according to the logical connectors.
   For example, v satisfies $F_1 \vee F_2$ if v satisfies F_1 or v satisfies F_2.

5. If F is $\exists x{:}A$ $(G(x_1, x_2, ..., x_n, x)$, then $<v_1, v_2, ..., v_n>$ satisifes F if there exists a constant c $\in$ DOM(A)
   $<v_1, v_2, ..., v_n, c>$ satisfies G.

6. If F is $\forall x{:}A$ $(G(x_1, x_2, ..., x_n, x)$, then $<v_1, v_2, ..., v_n>$ satisifes F if for every constant c $\in$ DOM(A)
   $<v_1, v_2, ..., v_n, c>$ satisfies G.