Database Applications (15-415)

Relational Calculus
Lecture 6, January 27, 2015

Mohammad Hammoud

جامعة كارنيجي ميلون في قطر
Carnegie Mellon University Qatar

Today…

- Last Session:
  - Relational Algebra
- Today's Session:
  - Relational calculus
    - Relational tuple calculus
- Announcements:
  - PS2 is now posted. Due on Feb 08, 2015 by midnight
  - PS1 grades are out
  - In the next recitation we will practice on relational algebra and calculus

Overview - Detailed

- Relational Tuple Calculus (RTC)
  - Why?
  - Details
  - Examples
  - Equivalence with relational algebra
  - ‘Safety’ of expressions

Motivation

- Question: What is the main “weakness” of relational algebra?

- Answer: Procedural
  - It describes the steps (i.e., ‘how’)
  - Still useful, especially for query optimization

Relational Calculus (in General)

- It describes what we want (*not how*)
- It has two equivalent flavors, 'tuple' and 'domain' calculus
  - We will only focus on relational 'tuple' calculus
- It is the basis for SQL and Query By Example (QBE)
- It is useful for proofs (see query optimization, later)

Relational Tuple Calculus (RTC)

- RTC is a subset of 'first order logic':

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

A "formula" that describes *t*

Give me tuples 't', satisfying predicate 'P'

- Examples:
  - Find all students: $\{t \mid t \in STUDENT\}$
  - Find all sailors with a rating above 7:

$$\{t \mid t \in Sailors \wedge t.rating > 7\}$$

Syntax of RTC Queries

- The allowed symbols:

$$\wedge, \quad \vee, \quad \neg, \quad \Rightarrow$$

$$>, \quad <, \quad =, \quad \neq, \quad \leq, \quad \geq,$$

$$(, \quad), \quad \in$$

- Quantifiers:

$$\forall, \quad \exists$$

Syntax of RTC Queries

- Atomic “formulas”:

$$t \in TABLE$$

$$t.attr\ \ op\ \ const$$

$$t.attr\ \ op\ \ s.attr$$

Where ***op*** is an operator in the set {<, >, =, ≤, ≥, ≠}

Syntax of RTC Queries

- A “formula” is:
  - Any atomic formula
  - If P1 and P2 are formulas, so are

$$\neg P1;\ \neg P2;\ P1 \wedge P2;\ P1 \vee P2;\ P1 \Rightarrow P2$$

  - If P(s) is a formula, so are

$$\exists s(P(s))$$

$$\forall s(P(s))$$

Basic Rules

- Reminders:
  - De Morgan: $P1 \wedge P2 \equiv \neg(\neg P1 \vee \neg P2)$
  - Implication: $P1 \Rightarrow P2 \equiv \neg P1 \vee P2$
  - Double Negation:

$$\forall s \in TABLE \quad (P(s)) \quad \equiv \quad \neg \exists s \in TABLE \quad (\neg P(s))$$

‘every human is mortal : no human is immortal’

A Mini University Database

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

Examples

- Find all student records

$$\{t \mid t \in STUDENT\}$$

output tuple

of type 'STUDENT'

Examples

- Find the student record with ssn=123

Examples

- Find the student record with ssn=123

$$\{t \mid t \in STUDENT \wedge t.ssn = 123\}$$

This is equivalent to the ‘Selection’ operator in Relational Algebra!

Examples

- Find the **name** of the student with ssn=123

$$\cancel{\{t \mid t \in STUDENT \wedge t.ssn = 123\}}$$

Will this work?

Examples

- Find the **name** of the student with ssn=123

$$\{t \mid \exists s \in STUDENT(s.ssn = 123 \wedge t.name = s.name)\}$$

't' has only one column

This is equivalent to the 'Projection' operator in Relational Algebra!

Examples

- Get records of both part time and full time students*

$$\{t \mid t \in FT\_STUDENT \quad \vee$$
$$t \in PT\_STUDENT\}$$

This is equivalent to the 'Union' operator in Relational Algebra!

* *Assume we maintain tables for PT_STUDENT and FT_STUDENT in our Mini University DB*

Examples

- Find students that are not staff*

$$\{t \mid t \in STUDENT \wedge$$
$$t \notin STAFF\}$$

This is equivalent to the 'Difference' operator in Relational Algebra!

* *Assume we maintain a table for STAFF in our Mini University DB and that STUDENT and STAFF are union-compatible*

Cartesian Product: A Reminder

- Assume MALE and FEMALE dog tables as follows:

MALE
<u>name</u>
spike
spot

x

FEMALE
<u>name</u>
lassie
shiba

=

M.Name	F.Name
spike	lassie
spike	shiba
spot	lassie
spot	shiba

This gives *all* possible couples!

Examples (Cont'd)

- Find all the pairs of (male, female) dogs

$$\{t \mid \exists m \in MALE \wedge$$
$$\exists f \in FEMALE$$
$$(t.m-name = m.name \wedge$$
$$t.f-name = f.name)\}$$

This is equivalent to the 'Cartesian Product' operator in Relational Algebra!

More Examples

- Find the names of students taking 15-415

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

2-way Join!

More Examples

- Find the names of students taking 15-415

$$
\begin{aligned}
&\{t \mid \exists s \in STUDENT \\
&\quad \wedge \exists e \in TAKES\ (\ s.ssn = e.ssn \wedge \\
&\qquad t.name = s.name \wedge \\
&\qquad e.c-id\ = 15-415)\}
\end{aligned}
$$

More Examples

- Find the names of students taking 15-415

$$\{t \mid \exists s \in STUDENT$$
$$\wedge \exists e \in TAKES\ (s.ssn = e.ssn \wedge$$
$$t.name = s.name \wedge$$
$$e.c-id = 15-415)\}$$

join

projection

selection

More Examples

- Find the names of students taking a 2-unit course

STUDENT		
Ssn	Name	Address
123	smith	main str
234	jones	forbes ave

CLASS		
c-id	c-name	units
15-413	s.e.	2
15-412	o.s.	2

TAKES		
SSN	c-id	grade
123	15-413	A
234	15-413	B

3-way Join!

More Examples

- Find the names of students taking a 2-unit course

$$
\begin{aligned}
&\{t \mid \exists s \in STUDENT \wedge \exists e \in TAKES \\
&\quad \exists c \in CLASS(s.ssn = e.ssn \wedge \\
&\quad\quad e.c - id = c.c - id \wedge \\
&\quad\quad t.name = s.name \wedge \\
&\quad\quad c.units = 2)\}
\end{aligned}
$$

join

projection

selection

What is the equivalence of this in Relational Algebra?

More on Joins

- Assume a Parent-Children (PC) table instance as follows:

PC	
p-id	**c-id**
Mary	Tom
Peter	Mary
John	Tom

PC	
p-id	**c-id**
Mary	Tom
Peter	Mary
John	Tom

- Who are Tom’s grandparent(s)? (*this is a self-join*)

More Join Examples

- Find Tom's grandparent(s)

$$
\begin{aligned}
&\{t \mid \exists p \in PC \wedge \exists q \in PC \\
&\quad (\ p.c-id = q.p-id \ \wedge \\
&\qquad p.p-id = t.p-id \wedge \\
&\qquad q.c-id = "Tom")\}
\end{aligned}
$$

What is the equivalence of this in Relational Algebra?

Harder Examples: DIVISION

- Find suppliers that shipped all the bad parts

SHIPMENT	
s#	p#
s1	p1
s2	p1
s1	p2
s3	p1
s5	p3

$\div$

BAD_P
p#
p1
p2

$=$

BAD_S
s#
s1

Harder Examples: DIVISION

- Find suppliers that shipped all the bad parts

$$\{t \mid \forall p(p \in BAD\_P \Rightarrow ($$
$$\exists s \in SHIPMENT($$
$$t.s\# = s.s\# \wedge$$
$$s.p\# = p.p\#)))\}$$

General Patterns

- There are three equivalent versions:
  1) If it is bad, he shipped it

$$\{t \mid \forall p(p \in BAD\_P \Rightarrow (P(t))\}$$

  2) Either it was good, or he shipped it

$$\{t \mid \forall p(p \notin BAD\_P \vee (P(t))\}$$

  3) There is no bad shipment that he missed

$$\{t \mid \neg \exists p(p \in BAD\_P \wedge (\neg P(t))\}$$

More on Division

- Find (SSNs of) students who are taking all the courses that ssn=123 is (and maybe even more)

One way to think about this:

Find students ‘s’ so that if 123 takes a course => so does ‘s’

More on Division

- Find (SSNs of) students who are taking all the courses that ssn=123 is (and maybe even more)

$$
\begin{aligned}
&\{o \mid \forall t((t \in TAKES \wedge t.ssn = 123) \Rightarrow \\
&\qquad \exists t1 \in TAKES(\\
&\qquad\quad t1.c - id = t.c - id \wedge \\
&\qquad\quad t1.ssn = o.ssn) \\
&\qquad))\}
\end{aligned}
$$

‘Proof’ of Equivalence

- Relational Algebra <-> RTC

But…

Safety of Expressions

- FORBIDDEN:

$$\cancel{\{t \mid t \notin STUDENT\}}$$

It has infinite output!!

- Instead, always use:

$$\{t \mid \ldots t \in SOME-TABLE\}$$

Summary

- The relational model has rigorously defined query languages — simple and powerful

- Relational algebra is more operational/procedural
  - Useful as internal representation for query evaluation plans

- Relational calculus is declarative
  - Users define queries in terms of what they want, not in terms of how to compute it

Summary

- Several ways of expressing a given query
  - A *query optimizer* should choose the most efficient version

- Algebra and "safe" calculus have same *expressive power*
  - leads to the notion of *relational completeness*

Next Class

SQL- Part I