Relational Calculus

Lecture 4**B**

Kathleen Durant
Northeastern University

Relational Calculus

- Relational Calculus is an alternative way for expressing queries
  - Main feature: specify what you want, not how to get it
  - Many equivalent algebra “implementations” possible for a given calculus expression
- In short: SQL query without aggregation = relational calculus expression
- Relational algebra expression is similar to program, describing what operations to perform in what order

What is Relational Calculus?

- It is a formal language based upon predicate calculus
- It allows you to express the set of tuples you want from a database using a formula

Relational Calculus

- Comes in two flavors: Tuple relational calculus (TRC) and Domain relational calculus (DRC)
- TRC: Variables range over (i.e., get bound to) tuples.
- DRC: Variables range over domain elements (= attribute values)
- Both TRC and DRC are subsets of first-order logic
  - We use some short-hand notation to simplify formulas
- Expressions in the calculus are called *formulas*
- Answer tuple = assignment of constants to variables that make the formula evaluate to true

Relational Calculus Formula

- Formula is recursively defined
  - Start with simple atomic formulas (getting tuples (or defining domain variables) from relations or making comparisons of values)
  - And build bigger and more complex formulas using the logical connectives.

Domain Relational Calculus

- Query has the form:
  - {<x1, x2,…, xn> | p(<x1, x2,…, xn>)}
  - Domain Variable – ranges over the values in the domain of some attribute or is a constant
  - Example: If the domain variable x_1 maps to attribute - Name char(20) then x_1 ranges over all strings that are 20 characters long
    - **Not just the strings values in the relation's attribute**
  - Answer includes all tuples <x1, x2,…, xn> that make the formula p(<x1, x2,…, xn>) true.

DRC Formulas

- Atomic Formulas
  1. $<x_1, x_2, \ldots, x_n> \in$ Relation
     - where Relation is a relation with n attributes
  2. X operation Y
  3. X operation *constant*
  - Where operation is an operator in the set $\{ <, >, =, \leq, \geq, \neq \}$
- Recursive definition of a Formula:
  1. An atomic formula
  2. $\neg p$, $p \wedge q$, $p \vee q$, where p and q are formulas
  3. $\exists X(p(X))$, where variable X is free in p(X)
  4. $\forall X(p(X))$, where variable X is free in p(X)
- The use of quantifiers $\exists X$ and $\forall X$ is said to bind X
  - A variable that is not bound is free.

Free and bound variables

- Let us revisit the definition of a query:
- $\{<x_1, x_2,\ldots, x_n> \mid p <x_1, x_2,\ldots, x_n>\}$
- Determine what assignment(s) to $<x_1, x_2,\ldots, x_n>$ make $\mid p <x_1, x_2,\ldots, x_n>$ true
- There is an important restriction:
  - The variables $x_1,\ldots, x_n$ that appear to the left of `|' must be the only free variables in the formula p<...>
  - All other variables in the query are bound

Quantifiers: ∀x and ∃x

- Variable x is said to be *subordinate* to the quantifier
  - You are restricting (or binding) the value of the variable x to set S
- The set S is known as the *range* of the quantifier
- ∀x predicate true for all elements in set S
- ∃x predicate true for at least 1 element in set S

Formula quantifier: $\forall$x

- $\forall$ is called the universal or "for all" quantifier because every tuple in "the universe of" tuples must make P(x) true to make the quantified formula true
- $\forall x\ (P(x))$ - is only true if P(x) is true for every x in the universe
- In our example we wrote:
  - $\forall \mathbf{x} \in$ **Boats(color = "Red")**
- It really means:
  - $\forall \mathbf{x}$ **((x** $\in$ **Boats)** $\Rightarrow$ **(color = "Red"))**
- $\Rightarrow$ logical implication What does it mean?
  - **A** $\Rightarrow$ **B** means that if A is true, B must be true
  - **Since A** $\Rightarrow$ **B** is the same as $\neg A \vee B$

$A \Rightarrow B$ is the same as $\neg A \vee B$

			B	$\neg A \vee B$
	A	$\neg A$	T	F
	T	F	T	F
	F	T	T	T

If A is TRUE, B has to be TRUE for $A \Rightarrow B$ to be TRUE

- If A is true and B is false, the implication evaluates to false.

If A is not true, B has no effect on the answer

- Since you have already satisfied the clause with $(\neg A)$
- The expression is always true.

Formula Quantifiers: ∃X

- ∃ is called the existential or "there exists" quantifier because any tuple that exists in "the universe of" tuples may make p(x) true to make the quantified formula true.
- $\exists X(p(X))$
  - Means p(X) is true for some X
  - There exists an X for which p(X) is true
- Shorthand notation
  - $\exists X \in Students(GPA = 3.2)$
- Real representation of the formula
  - $\exists X ((X \in Students) \land (GPA = 3.2))$
    - And logical operation as opposed to implication

Example : Domain RC

Table Students

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82
55516	Jones	jones@hist	Feb 11, 1992	2.98
55517	Ali	ali@math	Sep 22, 1989	3.11
55518	Smith	smith@math	Nov 30, 1991	3.32

- Find all students with the name 'Smith'

$$\{<I, N, L, D, G> \mid <I, N, L, D, G> \in Students \land N = \text{'Smith'}\}$$

- Can reference the relation in the expression as opposed to the attribute name

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82
55518	Smith	smith@math	Nov 30, 1991	3.32

Anatomy of a DRC query

$$\{\langle I, N, L, D, G \rangle \mid \langle I, N, L, D, G \rangle \in Students \wedge N = \text{‘Smith’}\}$$

- The condition <I, N,L,D,G > ϵ Students
  - ensures that the domain variables I, N, L,D, and G are bound to fields of the same Students tuple.
    - Maps it to an instance in the Students table
- The symbol ‘|’ from predicate calculus means ‘such that’
- The <I, N,L,D,G > to the left of `|’ (such that) says that every tuple <I, N,L,D,G > that satisfies N = ‘Smith’ is in the answer set.
- Calculus expresses answers to the query not operations like in relational algebra

DRC example with multiple predicates

- Find all students with the name ‘Smith’ and with a GPA > 3.5
- Just add another predicate { p Λ q }
- {<I, N,L,D,G > | <I, N,L,D,G > ϵ Students Λ N = ‘Smith’ Λ G > 3.5 }

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82
55516	Jones	jones@hist	Feb 11, 1992	2.98
55517	Ali	ali@math	Sep 22, 1989	3.11
55518	Smith	smith@math	Nov 30, 1991	3.32

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82

DRC: returning a field satisfying restrictions on multiple tables

- Find the name of all students that received a 'C' in a course
  - Retrieving data across 2 tables
- {<N> | $\exists$I, L, D, G($<$ I, N, L, D, G $>$ $\in$ *Students*

∧ ∃Ir,CN,CG(<Ir, CN, CG> ∈ Courses ∧ Ir = I ∧ CG = 'C'))}

SID	Name	Login	DoB	GPA
55515	Smith	smith@ccs	Jan 10,1990	3.82
55516	Jones	jones@hist	Feb 11, 1992	2.98
55517	Ali	ali@math	Sep 22, 1989	3.11
55518	Smith	smith@math	Nov 30, 1991	3.32

Sid	CId	Grade
55515	History 101	C
55516	Biology 220	A
55517	Anthro 320	B
55518	Music 101	A

Resulting Table

Name
Smith

Parsing a DRC query

Find the name of all students that received a ‘C’ in course

$\{<N> \mid$

$\exists I, L, D, G(< I, N, L, D, G > \in Students$

$\wedge\ \exists \mathbf{Ir,CN,CGr}(<Ir, CN, CG> \in Courses \wedge Ir = I \wedge CGr = \text{‘C’}))\}$

- Note the use of ∃ to find a tuple in Courses that `joins with’ the Students tuple under consideration
  - Student Id is the same value in both tables
  - Bound value represented via the variable Ir

Unsafe Queries, Expressive Queries

- It is possible to write syntactically correct calculus queries that have an *infinite* number of answers.
  - Such queries are called unsafe.
- Theorem: 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.

Relational Calculus: Summary

- Relational calculus is non-operational
  - Users define queries in terms of what they want, not in terms of how to compute it. (Declarativeness.)
- Algebra and safe calculus have the same expressive power, leading to the notion of relational completeness.
- Relational calculus had big influence on the design of SQL and Query-by-Example

Practice with relational algebra

Building up a Relational Algebra Function

Division Operation in RA A/B

- Given 2 relations A (courses) and B (students); A/B = let x, yA be two attributes in A and yB is an attribute in B with the same domain as the domain of yB
- A/B = {<x> such that for all <y> in B there exists <x ,y> an element of A = $\{< x > | \forall < y > \in B \; \exists < x, y > \in A\}$
- A/B contains all x tuples (courses) such that for every y tuple (students) in B, there is an xy tuple in A.
- • Or: If the set of y values (courses) associated with an x value (students) in A contains all y values in B, the x value is in A/B.
  - In general, x and y can be any lists of attributes
  - y is the list of fields in B, and x U y is the list of fields of A.
- Assume x = course id and y = student id - What is the query asking for?

The MEGA-STUDENT(s) someone who has taken all courses that are in the course table

Example of division

Table A

Student Id (x)	Course Id (y)
10	cs200
10	cs100
10	cs300
10	cs400
20	cs300
30	cs200
15	cs400
15	cs100
25	cs100
25	cs200

Instances of B

Course Id
cs200

Course Id
cs200
cs100

Course Id
cs100
Cs 200
cs300

Corresponding Instances of A/B

Student Id
10
30
25

Student Id
10
25

Student Id
10

Basic operations for Division

- Compute all x values in A that are not disqualified
  - How is a value disqualified?
  - If by attaching a y value from B, we obtain a tuple NOT in A
  - $\pi_x((\pi_x(A) \text{x}\ B) - A\ )$

- $\pi_x(A) - \ \pi_x((\pi_x(A) \text{x}\ B) - A\ )$

Step by step process of Division

B

Course Id
cs200

A

Student Id (x)	Course Id (y)
10	cs200
10	cs100
10	cs300
10	cs400
20	cs300
30	cs200
15	cs400
15	cs100
25	cs100
25	cs200

$(\pi_x(A) \text{x } B)$

10, cs200
20, cs200
30, cs200
15,cs200
25, cs200

$(\pi_x(A) \text{x } B)$ - A

20, cs200
15,cs200

$\pi_x((\pi_x(A) \text{x } B) - A)$

20
15

$\pi_x(A) - \pi_x((\pi_x(A) \text{x } B) - A)$

Student Id
10
30
25