Relational Calculus

Chapter 4, Part B

Why Is This Important?

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

Relational Calculus

- Comes in two flavors: Tuple relational calculus (TRC) and Domain relational calculus (DRC).
- Calculus has variables, constants, comparison operators, logical connectives and quantifiers.
  - 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.
- Expressions in the calculus are called *formulas*.
  - Answer tuple = assignment of constants to variables that make the formula evaluate to true.

Domain Relational Calculus

- Query has the form:
  {<x1, x2,…, xn> | p(<x1, x2,…, xn>)}
- Answer includes all tuples <x1, x2,…, xn> that make the formula p(<x1, x2,…, xn>) be true.
- Formula is recursively defined
  - Starting with simple atomic formulas (getting tuples from relations or making comparisons of values)
  - And building bigger and more complex formulas using the logical connectives.

DRC Formulas

- Atomic formula:
  - <x1, x2,…, xn> ∈ Rname, or X op Y, or X op *constant*
  - op is one of <, >, =, ≤, ≥, ≠
- Formula:
  - An atomic formula, or
  - ¬p, p∧q, p∨q, where p and q are formulas, or
  - ∃X(p(X)), where variable X is free in p(X), or
  - ∀X(p(X)), where variable X is free in p(X)
- The use of quantifiers ∃X and ∀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:

$$\left\{\left\langle x1, x2, \ldots, xn \right\rangle \mid p\left(\left\langle x1, x2, \ldots, xn \right\rangle\right)\right\}$$

- There is an important restriction:
  - The variables x1,..., xn that appear to the left of `|’ must be the only free variables in the formula p(...).

Find all sailors with a rating above 7

$$\{\langle I,N,T,A\rangle \mid \langle I,N,T,A\rangle \in \text{Sailors} \wedge T>7\}$$

- Condition <I,N,T,A>∈Sailors ensures that the domain variables I, N, T and A have to be fields of the same Sailors tuple.
- The term <I,N,T,A> to the left of `|' (which should be read as "such that") says that every tuple <I,N,T,A> that satisfies T>7 is in the answer set.
- Modify this query to answer:
  - Find sailors who are older than 18 or have a rating under 9, and are called 'Joe'.

Find sailors rated > 7 who have reserved boat #103

$$\{\langle I,N,T,A\rangle \mid \langle I,N,T,A\rangle \in \text{Sailors} \wedge T>7 \wedge$$
$$\exists Ir,Br,D\left(\langle Ir,Br,D\rangle \in \text{Reserves} \wedge Ir=I \wedge Br=103\right)\}$$

- We have used ∃ Ir,Br,D (...) as a shorthand for ∃Ir (∃Br (∃D (...)))
- Note the use of ∃ to find a tuple in Reserves that `joins with' the Sailors tuple under consideration.

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

$$\{\langle I,N,T,A\rangle \mid \langle I,N,T,A\rangle \in \text{Sailors} \wedge T>7 \wedge$$
$$\exists Ir,Br,D\left(\langle Ir,Br,D\rangle \in \text{Reserves} \wedge Ir=I \wedge\right.$$
$$\left.\exists B,BN,C\left(\langle B,BN,C\rangle \in \text{Boats} \wedge B=Br \wedge C='\text{red}'\right)\right)\}$$

- Observe how the parentheses control the scope of each quantifier's binding.
- This may look cumbersome, but with a good user interface, it can be very intuitive. (MS Access, QBE)

Find sailors who've reserved all boats

$$\{\langle I,N,T,A\rangle \mid \langle I,N,T,A\rangle \in \text{Sailors} \wedge$$
$$\forall B,BN,C\left(\neg\left(\langle B,BN,C\rangle \in \text{Boats}\right) \vee\right.$$
$$\left.\left(\exists Ir,Br,D\left(\langle Ir,Br,D\rangle \in \text{Reserves} \wedge I=Ir \wedge Br=B\right)\right)\right)\}$$

- Find all sailors I such that for each 3-tuple <B,BN,C> either it is not a tuple in Boats or there is a tuple in Reserves showing that sailor I has reserved it.

Find sailors who've reserved all boats (again)

$$\{\langle I,N,T,A\rangle \mid \langle I,N,T,A\rangle \in \text{Sailors} \wedge$$
$$\forall \langle B,BN,C\rangle \in \text{Boats}$$
$$\left(\exists \langle Ir,Br,D\rangle \in \text{Reserves}\left(I=Ir \wedge Br=B\right)\right)\}$$

- Simpler notation, same query. (Much clearer)
- To find sailors who've reserved all red boats:

$$\ldots\left(C\neq'\text{red}' \vee \exists \langle Ir,Br,D\rangle \in \text{Reserves}\left(I=Ir \wedge Br=B\right)\right)\}$$

Unsafe Queries, Expressive Power

- It is possible to write syntactically correct calculus queries that have an *infinite* number of answers.
  - Such queries are called unsafe.
  - E.g., {S | ¬(S∈Sailors)}
- 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.

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