

Relational Calculus

Chapter 4, Part B





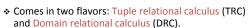
Why Is This Important?

- 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

2



Relational Calculus



- 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.

3

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:
 - $\langle x1, x2,..., xn \rangle \in Rname$, or X op Y, or X op constant
 - op is one of <, >, =, \le , \ge , \ne
- Formula:
 - An atomic formula, or
 - $\blacksquare \ \, \neg p,\, p \land q,\, p \lor q,$ where p and q are formulas, or
 - $\exists X(p(X))$, where variable X is free in p(X), or
 - $\forall X(p(X))$, 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:

$$\langle x1, x2, ..., xn \rangle | p(\langle x1, x2, ..., xn \rangle)$$

- * 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(...).

6

Find all sailors with a rating above 7

 $\langle I, N, T, A \rangle | \langle I, N, T, A \rangle \in Sailors \land T > 7$

- ❖ Condition <1,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

 $|\langle I, N, T, A \rangle| \langle I, N, T, A \rangle \in Sailors \land T > 7 \land$ $\exists Ir, Br, D |\langle Ir, Br, D \rangle \in Reserves \land Ir = I \land Br = 103||$

- ❖ 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.

7

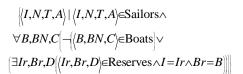
Find sailors rated > 7 who've reserved a red boat

 $\langle I, N, T, A \rangle | \langle I, N, T, A \rangle \in \text{Sailors} \land T > 7 \land$ $\exists Ir, Br, D \langle Ir, Br, D \rangle \in \text{Reserves} \land Ir = I \land$ $\exists B, BN, C \langle B, BN, C \rangle \in \text{Boats} \land B = Br \land C = \text{'red'}$

- 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)

0

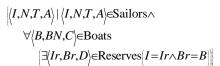
Find sailors who've reserved all boats



 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.

10

Find sailors who've reserved all boats (again)



- Simpler notation, same query. (Much clearer)
- * To find sailors who've reserved all red boats: $(C \neq \text{red} \lor \exists \langle Ir, Br, D \rangle \in \text{Reserves} [I = Ir \land Br = B]]$



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

13