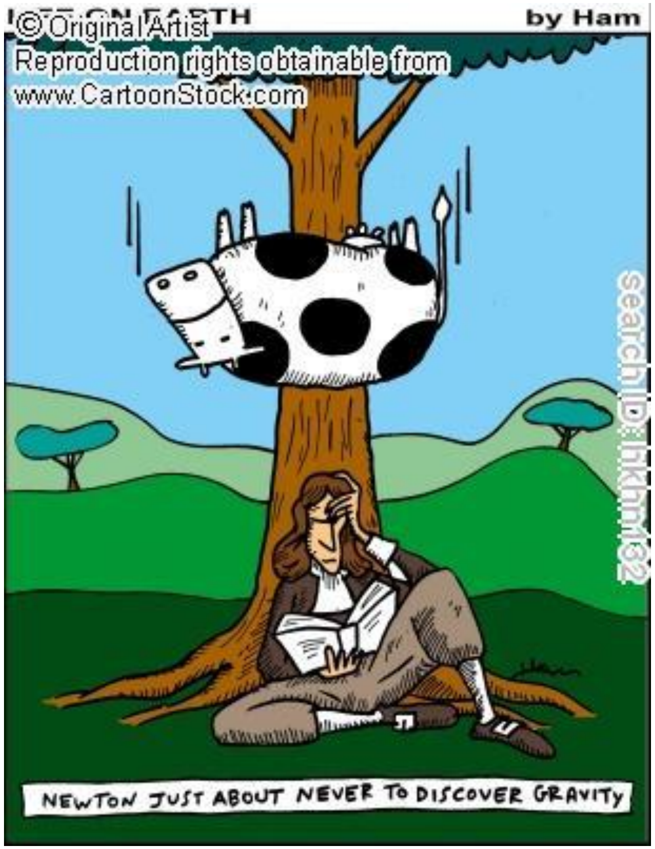
Relational Calculus

Chapter 4.3-4.5



Relational Calculus

- Comes in two flavors: *Tuple relational calculus* (TRC) and *Domain relational calculus* (DRC).
- Calculus has *variables, constants, comparison ops, logical connectives* and *quantifiers*.
  - *TRC*: Variables range over (i.e., get bound to) *tuples*.
  - *DRC*: Variables range over *domain elements* (= field values).
  - Both TRC and DRC are simple subsets of first-order logic.
- Expressions in the calculus are called *formulas with unbound formal variables*. An answer tuple is essentially an assignment of constants to these variables that make the formula evaluate to *true*.

A Fork in the Road

- TRC and DRC are semantically similar
- In TRC, tuples share an equal status as variables, and field referencing can be used to select tuple parts
- In DRC, formal variables are explicit
- In the book you will find extensive discussions and examples of TRC Queries (Sections 4.3.1) and a lesser treatment of DRC.
- To even things out, in this lecture I will focus on DRC examples

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>) *true*.

- *Formula* is recursively defined, starting with simple *atomic formulas* (getting tuples from relations or making comparisons of values), and building bigger and better formulas using the *logical connectives*.

DRC Formulas

- *Atomic formula:*
  - <x1,x2,…,xn> $\in$ *Rname*, or X *op* Y, or X *op* constant
  - *op* is one of $<, >, =, \leq, \geq, \neq$
- *Formula:*
  - an atomic formula, or
  - $\neg p, p \wedge q, p \vee q$, 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)

∃X(p(X)) is read as "*there exists a setting of the variable X such that p(X) is true*".
∀X(p(X)) is read as "*for all values of X, p(X) is true*"

Free and Bound Variables

- The use of quantifiers $\exists X$ and $\forall$ X in a formula is said to *bind* X.
  - A variable that is not bound is free.
- Let us revisit the definition of a query:

$$\{<x1,x2,\ldots,xn> \mid p(<x1,x2,\ldots,xn>)\}$$

- 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(...).

Examples

- Recall the example relations from last lecture

Sailors:

sid	sname	rating	age
22	Dustin	7	45.0
29	Brutus	1	33.0
31	Lubber	8	55.5
32	Andy	8	25.5
58	Rusty	10	35.0
64	Horatio	7	35.0
71	Zorba	10	16.0
74	Horatio	9	35.0
85	Art	3	25.5
95	Bob	3	63.5

Reservations:

sid	bid	day
22	101	10/10/98
22	102	10/10/98
22	103	10/8/98
22	104	10/7/98
31	102	11/10/98
31	103	11/6/98
31	104	11/12/98
64	101	9/5/98
64	102	9/8/98
74	103	9/8/98

Boats:

bid	bname	color
101	Interlake	blue
102	Interlake	red
103	Clipper	green
104	Marine	red

Find sailors with ratings > 7

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

- The condition $\langle I,N,T,A\rangle \in$ *Sailors* binds the domain variables *I, N, T* and *A* to fields of any Sailors tuple.
- The term, $\langle I,N,T,A\rangle$, to the left of '|' (which should be read as *such that*) says that every tuple, that satisfies $T > 7$ is in the answer.
- Modify this query to answer:
  - Find sailors who are older than 18 or have a rating under 9, and are called 'Joe'.

Same query using TRC

- Find all sailors with ratings above 7

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

- Note, here S is a tuple variable

$$\{X \mid S \in Sailors \wedge S.rating > 7 \wedge X.name = S.name \wedge X.age = S.age \}$$

- Here X is a tuple variable with 2 fields (name, age). This query implicitly specifies projection (π) and renaming (ρ) relational algebra operators

Sailors rated > 7 who reserved boat #103

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

- We have used $\exists\, Ir, Br, D(\ldots)$ as a shorthand for $\exists\, Ir(\exists\, Br(\exists\, D(\ldots)))$
- Note the use of $\exists$ to find a tuple in Reserves that 'joins with' ($\bowtie$) the Sailors tuples under consideration.

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

$$
\begin{aligned}
\{\langle I,N,T,A\rangle \mid\ & \langle I,N,T,A\rangle \in \mathit{Sailors} \wedge (T > 7) \wedge \\
& (\exists\, Ir, Br, D(\langle Ir, Br, D\rangle \in \mathit{Reserves} \wedge (Ir = I) \wedge \\
& \qquad (\exists\, B, Bn, C(\langle B, Bn, C\rangle \in \mathit{Boats} \wedge \\
& \qquad (B = Br) \wedge (C = \text{'red'}))))\}
\end{aligned}
$$

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

Names of all Sailors who have reserved boat 103

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailor$$
$$\wedge\ \exists Ir,Br,D(\langle Ir,Br,D\rangle \in Reserves \wedge Ir = I \wedge Br = 103))\}$$

- Note that only the *sname* field is retained in the answer and that only *N* is a free variable.
- A more compact version

$$\{\langle N\rangle \mid \exists I,T,A(\langle I,N,T,A\rangle \in Sailor$$
$$\wedge\ \exists D(\langle I,103,D\rangle \in Reserves))\}$$

Summary

- Relational algebra is operational. It explains how to execute a query. There may be many alternative executions that are equivalent.
- Relational calculus is non-operational. Users define queries in terms of what they want, not how to compute it. (Declarativeness.)
- Codd's insight: Relational algebra and "safe" relational calculus have same expressive power, leading to the notion of *relational completeness* and the foundation for databases.