



Relational Calculus



Relational Calculus

- ❖ Comes in two flavours: *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*. An answer tuple is essentially an assignment of constants to variables that make the formula evaluate to *true*.



Domain Relational Calculus

- ❖ *Query* has the form:

$$\left\{ \langle x_1, x_2, \dots, x_n \rangle \mid p(\langle x_1, x_2, \dots, x_n \rangle) \right\}$$

- ❖ *Answer* includes all tuples $\langle x_1, x_2, \dots, x_n \rangle$ that make the *formula* $p(\langle x_1, x_2, \dots, x_n \rangle)$ be *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:

- $\langle x_1, x_2, \dots, x_n \rangle \in Rname$, or $X \text{ op } Y$, or $X \text{ op } \text{constant}$
- **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)
- ❖ 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

- ❖ 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**:

$$\left\{ \langle x_1, x_2, \dots, x_n \rangle \mid p(\langle x_1, x_2, \dots, x_n \rangle) \right\}$$

- ❖ There is an important restriction: the variables **x_1, \dots, x_n** that appear to the left of `|` must be the ***only*** free variables in the formula $p(\dots)$.



Find all sailors with a rating above 7

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

- ❖ The condition $\langle I, N, T, A \rangle \in \text{Sailors}$ ensures that the domain variables I , N , T and A are bound to fields of the same 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 $\langle I, N, T, A \rangle$ 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'.



Find sailors rated > 7 who've reserved boat #103

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

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



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

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

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



Find sailors who've reserved all boats

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

- ❖ Find all sailors I such that for each 3-tuple $\langle B, BN, C \rangle$ 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!)

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

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

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



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., $\{S \mid \neg(S \in Sailors)\}$
- ❖ It is known that 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

- ❖ The relational model has rigorously defined query languages that are simple and powerful.
- ❖ Relational algebra is more operational; useful as internal representation for query evaluation plans.
- ❖ Relational calculus is non-operational, and users define queries in terms of what they want, not in terms of how to compute it. (*Declarativeness.*)
- ❖ Several ways of expressing a given query; a *query optimizer* should choose the most efficient version.
- ❖ Algebra and safe calculus have same *expressive power*, leading to the notion of *relational completeness*.