Relational Calculus

Chapter 4, Part B
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:
  \[ \{ \langle x_1, x_2, \ldots, x_n \rangle \mid p(\langle x_1, x_2, \ldots, x_n \rangle) \} \]

- **Answer** includes all tuples \( \langle x_1, x_2, \ldots, x_n \rangle \) that make the formula \( p(\langle x_1, x_2, \ldots, x_n \rangle) \) 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:**
  - \( \langle x_1, x_2, ..., x_n \rangle \in Rname \), or \( X op Y \), or \( X op \) constant
  - \( op \) is one of \( <,>,=,\leq,\geq,\neq \)

- **Formula:**
  - an atomic formula, or
  - \( \neg p, p \land q, p \lor q \), 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.
The use of quantifiers $\exists X$ and $\forall X$ in a formula is said to \textit{bind} $X$.

- A variable that is not bound is free.
- Let us revisit the definition of a query:

$$\{x_1, x_2, \ldots, x_n\} \mid p(x_1, x_2, \ldots, x_n)$$

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 $p(\ldots)$. 
Find all sailors with a rating above 7

\[ \{ \langle I, N, T, A \rangle | \langle I, N, T, A \rangle \in \text{Sailors} \land T > 7 \} \]

- 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 `\( \land \)` (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

\[
\{ (I,N,T,A) : (I,N,T,A) \in \text{Sailors} \land T > 7 \land \\
\exists \; Ir, Br, D \left( (Ir, Br, D) \in \text{Reserves} \land Ir = I \land Br = 103 \right) \}
\]

- We have used \( \exists \; Ir, Br, D \) (…) as a shorthand for \( \exists \; Ir \left( \exists \; Br \left( \exists \; D \left( \ldots \right) \right) \right) \)

- 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

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

- 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

\[ \{ \langle I,N,T,A \rangle \mid \langle I,N,T,A \rangle \in \text{Sailors} \land \forall B,BN,C \left( \neg (\langle B,BN,C \rangle \in \text{Boats}) \lor \right. \vphantom{\left[ \langle I,N,T,A \rangle \mid \langle I,N,T,A \rangle \in \text{Sailors} \land \forall B,BN,C \left( \neg (\langle B,BN,C \rangle \in \text{Boats} \right.} \vphantom{\left[ \langle I,N,T,A \rangle \mid \langle I,N,T,A \rangle \in \text{Sailors} \land \forall B,BN,C \left( \neg (\langle B,BN,C \rangle \in \text{Boats} \right.} \left. \right. \right. \vphantom{\left[ \langle I,N,T,A \rangle \mid \langle I,N,T,A \rangle \in \text{Sailors} \land \forall B,BN,C \left( \neg (\langle B,BN,C \rangle \in \text{Boats} \right.} \right. \right. \right. \right. \left. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. \right. } \exists Ir,Br,D \left( \langle Ir,Br,D \rangle \in \text{Reserves} \land I=Ir \land Br=B \right) \} \right) \right) \]

- 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!)

\[\{I, N, T, A\} | \{I, N, T, A\} \in \text{Sailors} \land \forall \{B, BN, C\} \in \text{Boats} \]

\[\exists \{Ir, Br, D\} \in \text{Reserves}(I = Ir \land Br = B)\]

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

\[\ldots \{C \neq \text{’red’} \lor \exists \{Ir, Br, D\} \in \text{Reserves}(I = Ir \land Br = B)\}\]
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., \[ \{ S \mid S \in \text{Sailors} \} \]

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

- Relational calculus is non-operational, and users define queries in terms of what they want, not in terms of how to compute it. (Declarativeness.)
- Algebra and safe calculus have same expressive power, leading to the notion of relational completeness.