Relational Calculus
Another Theoretical QL - 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*.
    - Like SQL.
  - **DRC**: Variables range over *domain elements* (= field values).
    - Like Query-By-Example (QBE)

- Both TRC and DRC are simple subsets of first-order logic.
  - We’ll focus on TRC here

- Expressions in the calculus are called *formulas*.

- **Answer tuple** is an assignment of constants to variables that make the formula evaluate to *true*. 
Tuple Relational Calculus

- **Query** has the form: \( \{ T \mid p(T) \} \)
  - \( p(T) \) denotes a formula in which tuple variable \( T \) appears.
- **Answer** is the set of all tuples \( T \) for which the formula \( p(T) \) evaluates to true.
- **Formula** is recursively defined:
  - start with simple *atomic formulas*
    - (get tuples from relations or make comparisons of values)
  - build bigger formulas using *logical connectives*. 
TRC Formulas

An Atomic formula is one of the following:

\[ R \in \text{Rel} \]

\[ R[a] \ op \ S[b] \quad \text{or} \quad R.a = S.b \]

\[ R[a] \ op \ \text{constant} \]

where \( op \) is one of \( \langle, >, =, \leq, \geq \),

A formula can be:

- an atomic formula
- \( \neg p, p \land q, p \lor q \) where \( p \) and \( q \) are formulas
- \( \exists R(p(R)) \) where variable \( R \) is a tuple variable
- \( \forall R(p(R)) \) where variable \( R \) is a tuple variable
Relational Calculus

Formula

\[ F \]

F (name, acct-no, Amt)

Result

F (smith, A101, 1000)

If TRUE

i.e., in DB
Free and Bound Variables

- Quantifiers $\exists X$ and $\forall X$ in a formula are said to bind $X$ in the formula.

  A variable that is not bound is free.

- Let us revisit the definition of a query:
  - $\{ T \mid p(T) \}$

- **Important restriction**
  - the variable $T$ that appears to the left of `|’ must be the only free variable in the formula $p(T)$.
  - in other words, all other tuple variables must be bound using a quantifier.
Example Schema

Sailors (sid, sname, age, rating)
Boats (bid, color)
Reserves (sid, bid)
Selection and Projection

- Find all sailors with rating above 7
  \[ \{ S \mid S \in Sailors \land S[\text{rating}] > 7 \} \]

- Modify this query to answer: Find sailors who are older than 18 or have a rating under 9, and are named ‘Bob’.
  \[ \{ S \mid \exists S1 \in Sailors (S1[\text{rating}] > 7 \land S[\text{sname}] = S1[\text{sname}] \land S[\text{age}] = S1[\text{age}]) \} \]

- Find names and ages of sailors with rating above 7.

  \[ \{ S \mid \exists S1 \in Sailors (S1[\text{rating}] > 7 \land S[\text{sname}] = S1[\text{sname}] \land S[\text{age}] = S1[\text{age}]) \} \]

- Note: \( S \) is a tuple variable with 2 attributes (i.e. \{S\} is a projection of \( Sailors \))
  - only 2 attributes are ever mentioned and \( S \) is never used to range over any relations in the query.
Find sailors and their rating for sailors rated > 7 who’ve reserved boat #103

\[ \{ S \mid S \in \text{Sailors} \land S[\text{rating}] > 7 \land \exists R \in \text{Reserves} \land (R[\text{sid}] = S[\text{sid}] \land R[\text{bid}] = 103) \} \]

Note the use of \( \exists \) to find a tuple in Reserves that `joins with` the Sailors tuple under consideration.
Joins (continued)

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

\{S \mid S \in \text{Sailors} \land S[\text{rating}] > 7 \land \exists R \in \text{Reserves} \ (R[\text{sid}] = S[\text{sid}] \land \exists B \in \text{Boats} \ (B[\text{bid}] = R[\text{bid}] \land B[\text{color}] = \text{‘red’}) )\}

- This may look cumbersome, but it’s not so different from SQL!
Division (makes more sense here???)

Find sailors who’ve reserved all boats

(hint, use ∀)

\{S \mid S\in\text{Sailors} \land 
\forall B \in \text{Boats} \ (\exists R \in \text{Reserves} 
(S[sid] = R[sid] 
\land B[bid] = R[bid]))\}

Find all sailors $S$ such that for all tuples $B$ in Boats there is a tuple in Reserves showing that sailor $S$ has reserved $B$. 
Unsafe Queries, Expressive Power

- ∃ syntactically correct calculus queries that have an infinite number of answers! **Unsafe** queries.
  - e.g., \( \{ S \mid \neg (S \in \text{Sailors}) \} \)
  - Solution???? Don’t do that!

- Expressive Power (Theorem due to Codd):
  - 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. (actually, SQL is more powerful, as we will see…)
Find the names of customers w/ loans at the Perry branch. Answer has form \( \{ t \mid P(t) \} \).

Strategy for determining \( P(t) \):

1. What tables are involved?
   
   borrower \((s)\), loan \((u)\)
   
2. What are the conditions?
   
   (a) Projection: \( t[cname] = s[cname] \)
   (b) Join: \( s[lno] = u[lno] \)
   (c) Selection: \( u[bname] = \text{“Perry”} \)
Find the names of customers w/ loans at the Perry branch.

A. \( \{ t \mid \exists s \in \text{borrower} \ (P(t,s)) \} \) such that:

\[
P(t,s) \equiv t [\text{cname}] = s [\text{cname}] \land \exists u \in \text{loan} \ (Q(t,s,u))
\]
\[
Q(t,s,u) \equiv s [\text{lno}] = u [\text{lno}] \land u [\text{bname}] = \text{“Perry”}
\]

OR unfolded version (either is ok)

\[
\{ t \mid \exists s \in \text{borrower} \ (t [\text{cname}] = s [\text{cname}] \land \\
\exists u \in \text{loan} \ (s [\text{lno}] = u [\text{lno}] \land u [\text{bname}] = \text{“Perry”}))\}\]