Outline

- Relational Model
  - Formal Query Languages
    - Relational Algebra
    - Relational Calculus
    - Language Theory
Query Languages

- The three languages we consider:
  - Relational Algebra (RA)
  - Relational Calculus (RC)
  - Structured Query Language (SQL)
Find sailors rated > 7 who have reserved boat #103

Relational Algebra:

\[ \pi_{\text{sname}}((\sigma_{\text{bid}=103}(\text{Reserves} \bowtie (\sigma_{\text{rating}>7}(\text{Sailors})))) \]

Relational Calculus:

\[ \{X_{\text{sname}} | \exists X_{\text{sid}}, X_{\text{rating}}, X_{\text{age}} \text{ Sailors}(X_{\text{sid}}, X_{\text{sname}}, X_{\text{rating}}, X_{\text{age}}) \land X_{\text{rating}}>7 \land \exists X_{\text{bid}}, X_{\text{day}} \text{ Reserves}(X_{\text{sid}}, X_{\text{bid}}, X_{\text{day}}) \land X_{\text{bid}}=103 \} \]

SQL:

```
SELECT  sname
FROM     Sailors S, Reserves R
WHERE   S.sid=R.sid and s.rating>7 and R.bid = '103';
```
Unsafe Queries, Expressive Power

- *Unsafe* queries in calculus:
  - some queries can have an infinite number of answers.
  - e.g., \( S | \neg (S \in \text{Sailors}) \)

- Equivalence between RA and Safe RC

**Theorem:** every query that can be expressed in relational algebra can be expressed as a safe query in relational calculus; the converse is also true.
Query Language Classes

Recursive Queries

First Order Queries

Conjunctive Queries

Algebra Logic SQL

RA (safe) RC SFW +
UNION, EXCEPT
no aggregation
Query Language Classes

Recursive Queries

First Order Queries

Conjunctive Queries

<table>
<thead>
<tr>
<th></th>
<th>Algebra</th>
<th>Logic</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA</td>
<td>(safe) RC</td>
<td>SFW + UNION, EXCEPT no aggregation</td>
<td></td>
</tr>
<tr>
<td>RA: σ,π,×</td>
<td>Single datalog rule</td>
<td>S^{d}FW no aggregation</td>
<td></td>
</tr>
</tbody>
</table>
Conjunctive Queries (CQ)

- A **subset** of FO queries (i.e., less expressive).
- CQs have “better” theoretical properties than arbitrary queries.
- Query optimizer handles CQs the best (it tries to)
  - flatten a nested query to a single CQ
  - break a large query into many CQs
CQ in Rule-based (Datalog) Notation

- **R**: Extensional database (EDB) -- stored
- **P**: Intentional database (IDB) -- computed
Find sailors who have reserved boat #103

\[ P(x, z) \leftarrow R_1(x, y) \& R_2(y, z) \]

\[ P(X_{sname}) \leftarrow \text{Sailors}(X_{sid}, X_{sname}, X_{rating}, X_{age}) \]
\& \text{Reserves}(X_{sid}, 103, X_{day}) \]
Properties of CQs

- **Satisfiability**
  - A query Q is *satisfiable* if there exists at least one database instance D such that Q(D) is non-empty.
  - Theorem: *Every CQ is satisfiable.*

- **Monotonicity**
  - A query Q is *monotonic* if for two database instances D1 and D2, D1 ⊆ D2 implies Q(D1) ⊆ Q(D2).
  - Theorem: *Every CQ is monotonic.*
Consequence of Monotonicity

- Are these queries monotonic?
- Are they in the class of conjunctive queries?
Beyond First-Order Queries

Recursive Queries

First Order Queries

Conjunctive Queries

Algebra

Logic

SQL

RA

(safe) RC

SFW +
UNION, EXCEPT
no aggregation

RA:
\( \sigma, \pi, \times \)

Single
data
golog
rule

S\text{dFW}
no aggregation
Limitation of FO Queries

- Let $D = \{E(x,y)\}$ represent a graph
- Query $\text{path}(x, z) =$
  - all $x, z$ such that there is a path from $x$ to $z$.

Inexpressibility

Theorem: Query $\text{path}(x, z)$ cannot be expressed in First Order (FO) queries.
Find all of Mary’s ancestors

<table>
<thead>
<tr>
<th>Parent</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Joe</td>
</tr>
<tr>
<td>Joe</td>
<td>Alice</td>
</tr>
<tr>
<td>Joe</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>Mary</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

- Can we write a query in SQL?
SQL with Recursion

Relation to be computed recursively

WITH RECURSIVE Ancestor(anc, desc) AS

((SELECT parent AS anc, child AS desc
FROM ParentOf)
UNION
((SELECT A.anc, P.child AS desc
FROM Ancestor A,
ParentOf P
WHERE A.desc = P.parent)

SELECT anc
FROM Ancestor
WHERE desc = 'Mary';
## Recursive Computation

### Ancestor

<table>
<thead>
<tr>
<th>Anc</th>
<th>Desc</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Joe</td>
</tr>
<tr>
<td>Joe</td>
<td>Alice</td>
</tr>
<tr>
<td>Joe</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>Mary</td>
</tr>
<tr>
<td>Mike</td>
<td>Alice</td>
</tr>
<tr>
<td>Mike</td>
<td>Bob</td>
</tr>
<tr>
<td>Joe</td>
<td>Mary</td>
</tr>
<tr>
<td>Mike</td>
<td>Mary</td>
</tr>
</tbody>
</table>

### ParentOf

<table>
<thead>
<tr>
<th>Parent</th>
<th>Child</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Joe</td>
</tr>
<tr>
<td>Joe</td>
<td>Alice</td>
</tr>
<tr>
<td>Joe</td>
<td>Bob</td>
</tr>
<tr>
<td>Alice</td>
<td>Mary</td>
</tr>
</tbody>
</table>

### Base case:

- Mike → Joe
- Joe → Alice
- Joe → Bob
- Alice → Mary

### Iter 1:

- Mike → Alice
- Mike → Bob
- Joe → Mary

### Iter 2:

- Mike → Mary

### Iter 3:

- { } reached fixed point

### Query result:

- Alice → Mary
- Joe → Mary
- Mike → Mary

---

---
Recursion in Datalog

\[
\begin{align*}
\text{Ancestor}(x, y) & \leftarrow \text{ParentOf}(x, y) \\
\text{Ancestor}(x, z) & \leftarrow \text{Ancestor}(x, y) \; \& \; \text{ParentOf}(y, z)
\end{align*}
\]

Use of IDB in Body

Implicit UNION

\[
\text{AncestorOfMary}(x) \leftarrow \text{Ancestor}(x, \text{‘Mary’})
\]
## A More Complete Picture

<table>
<thead>
<tr>
<th>Recursive Queries</th>
<th>Algebra</th>
<th>Logic</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA + Fixed Point operator</td>
<td>Datalog (recursion)</td>
<td>Full SQL (recursion)</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>First Order Queries</th>
<th>Algebra</th>
<th>Logic</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA</td>
<td>(safe) RC</td>
<td>SFW + UNION, EXCEPT no aggregation</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Conjunctive Queries</th>
<th>Algebra</th>
<th>Logic</th>
<th>SQL</th>
</tr>
</thead>
<tbody>
<tr>
<td>RA: $\sigma,\pi,\times$</td>
<td>Single Datalog rule</td>
<td>S$^d$FW no aggregation</td>
<td></td>
</tr>
</tbody>
</table>