The Relational Language

Todd S. Bacastow
January 2004
Relational Query Languages

- Query languages
  - Allow manipulation and retrieval of data from a database.

- Query Languages
  - QLs not intended to be used for complex calculations.
  - QLs support easy, efficient access to large data sets.

- Two types of languages
  - Formal (mathematical) query language (ch 4)
    - Relational Algebra
    - Relational Calculus
  - Real (or operational) language --- SQL (ch 5)
Formal Relational Query Languages

Two mathematical Query Languages form the basis for “real” languages (e.g. SQL), and for implementation:

✓ **Relational Algebra**: More operational, very useful for representing execution plans.

✓ **Relational Calculus**: Lets users describe what they want, rather than how to compute it.
Relational Calculus

- Comes in two flavours:
  - *Tuple relational calculus* (TRC)
  - *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).

- 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*. 
Relational Algebra

- Basic operations:
  - **Selection** (\( \sigma \)) Selects a subset of rows from relation.
  - **Projection** (\( \pi \)) Deletes unwanted columns from relation.
  - **Cross-product (Cartesian product)** (\( \times \)) Allows us to combine two relations and returns all possible combinations.
  - **Union** (U) All tuples in reln. 1 and in reln. 2.
  - **Intersection** (\( \cap \)) Only tuples common the reln. 1 and reln 2.
  - **Set-difference** (\( \setminus \)) Tuples in relation 1, but not in reln. 2.
  - **Join** (\( \bowtie \)) Cross-product and a selection.

- Additional operations:
  - Intersection, division: Not essential, but useful.
  - Since each operation returns a relation, **operations can be composed.**
Relational Algebra Operations

Columns!!

Glue together all possible combinations

Rows!!

Everything in reln 1 plus reln 2

(a) Selection

(b) Projection

(c) Cartesian product

(d) Union

(e) Intersection

(f) Set difference

\[
R 
\times S
\]

\[
\begin{bmatrix}
a \\
b \\
1 \\
2 \\
3 \\
\end{bmatrix}
\times
\begin{bmatrix}
a \\
b \\
1 \\
2 \\
3 \\
\end{bmatrix}
= 
\begin{bmatrix}
a \\
b \\
1 \\
2 \\
3 \\
\end{bmatrix}
\]
Consider the following database schema

- The schema:
  
  Store (SName, City)
  PriceList (UPC, SName, Price)
  Item (UPC, IName, Warranty)
The tables

<table>
<thead>
<tr>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>SName</strong></td>
</tr>
<tr>
<td>Bloomingbirds</td>
</tr>
<tr>
<td>Harrods</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PriceList</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UPC</strong></td>
</tr>
<tr>
<td>12345</td>
</tr>
<tr>
<td>98765</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Item</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>UPC</strong></td>
</tr>
<tr>
<td>12345</td>
</tr>
<tr>
<td>98765</td>
</tr>
</tbody>
</table>
Example

- Write a relational algebra query to find the names of all stores that sell items with warranty equal to 24 months.

\[ \pi_{\text{StoreName}}(\sigma_{\text{Warranty}=24}(\text{PriceList} \bowtie \text{Item})) \]

1. Join PriceList and Item
2. Select those with a warranty equal to 24 months
3. Return the store name
Join PriceList and Item

\[(\text{PriceList} \bowtie \text{Item})\]

**PriceList**

<table>
<thead>
<tr>
<th>UPC</th>
<th>SName</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>Bloomingbirds</td>
<td>$300</td>
</tr>
<tr>
<td>98765</td>
<td>Harrods</td>
<td>$595</td>
</tr>
</tbody>
</table>

**Item**

<table>
<thead>
<tr>
<th>UPC</th>
<th>IName</th>
<th>Warranty</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>CPU</td>
<td>24 mos</td>
</tr>
<tr>
<td>98765</td>
<td>Airplane</td>
<td>60 mos</td>
</tr>
</tbody>
</table>

**Item_Price**

<table>
<thead>
<tr>
<th>UPC</th>
<th>IName</th>
<th>Warranty</th>
<th>SName</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>CPU</td>
<td>24 mos</td>
<td>Bloomingbirds</td>
<td>$300</td>
</tr>
<tr>
<td>98765</td>
<td>Airplane</td>
<td>60 mos</td>
<td>Harrods</td>
<td>$595</td>
</tr>
</tbody>
</table>
Select all tuples where the warranty = 24 months

\[(\sigma_{\text{Warranty}=24}(\text{PriceList} \bowtie \text{Item}))\]

<table>
<thead>
<tr>
<th>UPC</th>
<th>IName</th>
<th>Warranty</th>
<th>SName</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>12345</td>
<td>CPU</td>
<td>24 mos</td>
<td>Bloomingbirds</td>
<td>$300</td>
</tr>
</tbody>
</table>
Select the column with the store names

\[ \pi_{\text{SName}}(\sigma_{\text{Warranty}=24}((\text{PriceList} \bowtie_\text{Item}))) \]

<table>
<thead>
<tr>
<th>SName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bloomingbirds</td>
</tr>
</tbody>
</table>
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.
- SQL is an implementation of relational algebra.
- There may be several ways of expressing a given query.