Course Administration

- Assignment #1 is due today (outside 336/338).
- Assignment #2 will be out on the home webpage.
  - It is due two weeks from today.
- Course slides will now have black background
  - Printer friendly: set the printing color to “black-white”
- Next week reading:
  - Chapter 5 SQL
Vision 2010 (NTT DoCoMo)

Reflection: DB design

• Step 1: Requirements Analysis
  – What data to store in the database?
• Step 2: Conceptual Database Design
  – Come up with the design: Entity-Relation (ER) model
  – Sketch the design with entity-relationship diagrams
• Step 3: Logical Database Design
  – Implement the design: relational data model
  – Map ER diagrams to relational tables
• What’s next (after creating all these nice tables)?
What’s next?

• How to ask questions about the [relational] database?
  – How much money in account XYZ?
  – Who are valuable customers [$\sum$ deposits > 1M]?
  – Find the better 3-4 combination in MLB that is better (on-based percentage) than Ramirez and Ortiz (Boston Red Sox).

• Two query languages
  – SQL [CH5] : a Real query Language

Relational Algebra

Chapter 4.1 – 4.2
Relational Query Languages

• What are query languages?
  – For asking questions about the database
• Relational Algebra
  – Mathematical Query Languages form the basis for “real” languages (e.g. SQL) and for implementation.
  – A relational query is composed using a small set of operators (π, σ, ∞, X, …)
    • Like +, -, *, / operators in algebra

Preliminaries

• A query is applied to table(s), and the result of a query is also a table.
  – Schema of input table(s) for a query is fixed.
  – Schema for the result of a given query is also fixed! Determined by definition of query language constructs.
• Example of a relational algebra expression:
  – Find the names of sailors who have reserved boat 103
    \[ \pi_{\text{sname}}(\sigma_{\text{bid} = 103} \text{Reserves}) \bowtie \text{Sailors} \]
Example Tables

- Sailors and Reserves are tables.
- Can refer to the fields by their positions or names.
- Assume names of fields in the result table are inherited from names of fields in input tables.

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>guppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

Relational Algebra

- Basic relational algebra operators:
  - Selection (σ, pronounced sigma): Select a subset of rows from a table.
  - Projection (π): Delete unwanted columns from a table.
  - Cross-product (X): Combine two tables.
  - Set-difference (\(-\)): Tuples in table 1, but not in table 2.
  - Union (U): Tuples in tables 1 or 2.
- Each operator takes one or two input table(s), and returns one table.
Relational Algebra (more)

- Additional relational algebra operators:
  - Intersection (\( \cap \)): Tuples in tables 1 and 2.
  - Join (\( \bowtie \)): conditional cross product
  - Division (\( / \)): will explain later
  - Renaming (\( p \), pronounced "row")
- Operations can be composed to form a very complex query
  \[
  \pi_{sid} (\sigma_{\text{age} > 20} \text{Sailors}) - \\
  \pi_{sid} ((\sigma_{\text{color} = \text{'red'}} \text{Boats}) \bowtie \text{Reserves} \bowtie \text{Sailors})
  \]

Relational Operators

- Projection
- Selection
- Union
- Intersection
- Set difference
- Cross product
- Rename operator
- Join
- Division
Projection

- Delete attributes not in projection list.
- Duplicates eliminated
- Find ages of sailors in S2
- Find names of sailors in S2

\[
\pi_{\text{name}, \text{rating}}(S2)
\]

\[
\left| \begin{array}{ccc}
\text{name} & \text{rating} \\
\text{yuppy} & 9 \\
\text{lubber} & 8 \\
\text{guppy} & 5 \\
\text{rusty} & 10 \\
\end{array} \right|
\]

\[
\pi_{\text{age}}(S2)
\]

\[
\left| \begin{array}{c}
\text{age} \\
35.0 \\
55.5 \\
\end{array} \right|
\]

Relational Operators

- Projection
- Selection
- Union
- Intersection
- Set difference
- Cross product

- Rename operator
- Join
- Division
Selection

- Selects rows satisfying selection condition.
  - with duplicate removal
- Result table can be fed into other operations
- Find (names, ratings) of sailors whose ratings are greater than 8.
- Find names of sailors whose ages are greater than 40.

\[
\sigma_{\text{rating} > 8}(S2)
\]

\[
\pi_{\text{name}, \text{rating}}(\sigma_{\text{rating} > 8}(S2))
\]

Relational Operators

- Projection
- Selection
- Union
- Intersection
- Set difference
- Cross product
- Rename operator
- Join
- Division
### Union

- Take two input tables, which must be union-compatible:
  - Same number of fields.
  - 'Corresponding' fields have the same type.
- What is the schema of result?
- Find sailors in S1 or S2.

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>28</td>
<td>uppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

### Intersection

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
<tr>
<td>28</td>
<td>uppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>
Set-Difference

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>Dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>28</td>
<td>Yuppy</td>
<td>9</td>
<td>35.0</td>
</tr>
<tr>
<td>31</td>
<td>Lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>44</td>
<td>Guppy</td>
<td>5</td>
<td>35.0</td>
</tr>
<tr>
<td>58</td>
<td>Rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

\[ S1 - S2 \]

Relational Operators

- Projection
- Selection
- Union
- Intersection
- Set difference
- Cross product
- Rename operator
- Join
- Division
Cross-Product

- Each row of S1 is paired with each row of R1.
- Result schema has one field per field of S1 and R1, with field names 'inherited' if possible.
  - Conflict: Both S1 and R1 have a field called sid.

**S1**

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

**S1 x R1**

<table>
<thead>
<tr>
<th>(sid)</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
<th>(sid)</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
<td>31</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>

Renaming operator: $\rho(C(1\rightarrow sid1,5\rightarrow sid2), S1\times R1)$

---

Condition Joins

$S1\bowtie S1.sid < R1.sid R1$  
$R\bowtie_c S = \sigma_c (R \times S)$

**R1**

<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>

**S1**

<table>
<thead>
<tr>
<th>sid</th>
<th>sname</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>

- Cross-product, followed by a selection
- Result schema same as that of cross-product.
- Fewer tuples than cross-product, reduce tuples not meeting the condition.
Equi-Joins

- A special case of condition join where the condition $c$ contains only equalities.
- Result schema similar to cross-product, but only one copy of fields for which equality is specified.
- Natural Join ($\bowtie$): Equi-join on all common fields.

```
<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>
```

Example of Join

- Find the names of sailors who have reserved at least a boat.
  - $\pi_{\text{sname}}(R1 \bowtie S1)$
  - $\pi_{\text{sname}}(R1 \bowtie R1.\text{sid} = S1.\text{sid} S1)$
  - $\pi_{\text{sname}}(R1 \bowtie \text{sid} S1)$

```
<table>
<thead>
<tr>
<th>sid</th>
<th>bid</th>
<th>day</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>101</td>
<td>10/10/96</td>
</tr>
<tr>
<td>58</td>
<td>103</td>
<td>11/12/96</td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>sid</th>
<th>name</th>
<th>rating</th>
<th>age</th>
</tr>
</thead>
<tbody>
<tr>
<td>22</td>
<td>dustin</td>
<td>7</td>
<td>45.0</td>
</tr>
<tr>
<td>31</td>
<td>lubber</td>
<td>8</td>
<td>55.5</td>
</tr>
<tr>
<td>58</td>
<td>rusty</td>
<td>10</td>
<td>35.0</td>
</tr>
</tbody>
</table>
```
Relational Operators

• Projection
• Selection
• Union
• Intersection
• Set difference
• Cross product
• Rename operator
• Join
• Division

Division: analogy to integer division

• Two integers: $A$, $B$
  – $A/B = Q$, $Q$ is the largest integer such that $Q \times B \leq A$
• How about relational tables $A$, $B$?
  – $A/B = Q$, $Q$ is the largest table such that $Q \times B \leq A$
• Look at $Q \times B$ in $A$
  – $Q$ must be a subset of attributes in $A$
  – $Q$'s attributes + $B$'s attributes = $A$'s attributes
  – $A$'s tuples must contain all pairings $<q \text{ in } Q, b \text{ in } B>$
Division

- **Reserves**(sailor\_name, boat\_id); **Boats**(boat\_id)
  - Useful for expressing queries like:
    
    *Find sailors who have reserved all boats* $\Rightarrow$ **Reserves** / **Boats**

- Let $A$ have 2 fields, $x$ and $y$; $B$ have only field $y$:
  - $A/B = \{\langle x \rangle | \exists \langle x, y \rangle \in A \land \langle y \rangle \in B\}$
  - $A(x\_y)/B(y)$ contains all $x$ tuples (sailor\_name) such that for every $y$ tuple
    (boat\_id) in $B$, there is an $xy$ tuple in $A$.

**Examples of Division $A/B$**

- $X$ $Y$
  - $X1$ $Y1$
  - $X1$ $Y2$
  - $X1$ $Y3$
  - $X1$ $Y4$
  - $X2$ $Y1$
  - $X2$ $Y2$
  - $X3$ $Y2$
  - $X4$ $Y2$
  - $X4$ $Y4$

- $Y$
  - $Y1$
  - $Y2$
  - $Y4$

- $A/B1$
  - $X$
    - $X1$
    - $X2$
    - $X3$
    - $X4$

- $B1$
Expressing A/B Using Basic Operators

- Idea: For \( A(xy)/B(y) \), compute all \( x \) values that are not disqualified by some \( y \) value in \( B \).
  - \( x \) value is disqualified if by attaching \( y \) value from \( B \), we obtain an \( xy \) tuple that is not in \( A \).
  - 1) Iterate through each \( x \) value
  - 2) Check: combined with each \( y \) value, \( xy \) in \( A \)? If not, disqualify.

- Disqualified \( x \) values:

\[
A/B = \pi_x(A) - \pi_x((A) \times B)
\]
Practices with Relational Operators

(Q1) Find names of sailors who’ve reserved boat #103

Reserves(sid, bid, day)
Sailors(sid, sname, rating, age)

• Solution 1:
  \[ \pi_{\text{sname}}(\sigma_{\text{bid} = 103}(\text{Reserves} \bowtie \text{Sailors})) \]

• Solution 2 (more efficient)
  \[ \pi_{\text{sname}}(\sigma_{\text{bid} = 103}(\text{Reserves} \bowtie \text{Sailors})) \]

• Solution 3 (using rename operator)
  \[ P(\text{Temp1}, \sigma_{\text{bid} = 103}\text{Reserves}) \]
  \[ P(\text{Temp2}, \text{Temp1} \bowtie \text{Sailors}) \]
  \[ \pi_{\text{sname}}(\text{Temp2}) \]
(Q2) Find names of sailors who’ve reserved a red boat

\[
\text{Reserves}(\text{sid}, \text{bid}, \text{day}) \quad \text{Sailors}(\text{sid}, \text{sname}, \text{rating}, \text{age}) \quad \text{Boats}(\text{bid}, \text{bname}, \text{color})
\]

• Solution?

\[
\pi_{\text{sname}}(\sigma_{\text{color} = 'red'} \text{Boats}) \bowtie \text{Reserves} \bowtie \text{Sailors}
\]

• A more efficient solution (# of disk access)?

\[
\pi_{\text{sname}}(\pi_{\text{sid}}(\sigma_{\text{color} = 'red'} \text{Boats}) \bowtie \text{Reserves} \bowtie \text{Sailors})
\]

A query optimizer can find this, given the first solution!

(Q3) Find the colors of boats reserved by Lubber

\[
\text{Reserves}(\text{sid}, \text{bid}, \text{day}) \quad \text{Sailors}(\text{sid}, \text{sname}, \text{rating}, \text{age}) \quad \text{Boats}(\text{bid}, \text{bname}, \text{color})
\]

• Solution?

\[
\pi_{\text{color}}(\sigma_{\text{sname} = 'Lubber'} \text{Sailor}) \bowtie \text{Reserves} \bowtie \text{Boats}
\]
(Q4) Find the names of sailors who have reserved at least one boat

\[
\begin{align*}
\text{Reserves}(\text{sid, bid, day}) \\
\text{Sailors}(\text{sid, sname, rating, age}) \\
\text{Boats}(\text{bid, bname, color})
\end{align*}
\]

- Solution?

\[
\pi_{\text{sname}}(\text{Sailor} \bowtie \text{Reserves})
\]

(Q5) Find the names of sailors who have reserved a red or a green boat

\[
\begin{align*}
\text{Reserves}(\text{sid, bid, day}) \\
\text{Sailors}(\text{sid, sname, rating, age}) \\
\text{Boats}(\text{bid, bname, color})
\end{align*}
\]

- Solution?

\[
\pi_{\text{sname}}(\sigma_{\text{color} = \text{red} \text{ or color = green}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors})
\]
(Q6) Find the names of sailors who have reserved a red and a green boat

\[ \begin{align*}
\text{Reserves}(\text{sid}, \text{bid}, \text{day}) \\
\text{Sailors}(\text{sid}, \text{name}, \text{rating}, \text{age}) \\
\text{Boats}(\text{bid}, \text{name}, \text{color})
\end{align*} \]

• Wrong solution:

\[ \pi_{\text{name}}(\sigma_{\text{color}= \text{red} \text{ and color = green}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors}) \]

• What’s wrong with the above?

• A correct solution?

\[ \pi_{\text{name}}(\sigma_{\text{color}= \text{red}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors}) \cap \pi_{\text{name}}(\sigma_{\text{color}= \text{green}} \text{Boats} \bowtie \text{Reserves} \bowtie \text{Sailors}) \]

(Q7) Find the names of sailors who have reserved at least two different boats

\[ \begin{align*}
\text{Reserves}(\text{sid}, \text{bid}, \text{day}) \\
\text{Sailors}(\text{sid}, \text{name}, \text{rating}, \text{age}) \\
\text{Boats}(\text{bid}, \text{name}, \text{color})
\end{align*} \]

• Strategy?

  – Join a table (sid, bid, name): sailors reserving at least one boat
  – Cross-product the table with itself
  – Select sailors with two different boats reserved

\[ P (\text{Reservations}, C(1\rightarrow\text{sid1}, 2\rightarrow\text{sid2}, 3\rightarrow\text{bid1}, 4\rightarrow\text{bid2})) \]

\[ \pi_{\text{sid}, \text{name}, \text{bid}}(\text{Sailors} \bowtie \text{Reserves}) \]

\[ \pi_{\text{name}}(\sigma_{(\text{sid1} = \text{sid2}) \& (\text{bid}1 \neq \text{bid}2)} \text{Reservations} \times \text{Reservations}) \]
(Q8) Find the sids of sailors with age over 20 who have not reserved a red boat

\[ \pi_{\text{sid}} (\sigma_{\text{age}>20} \text{Sailors}) - \pi_{\text{sid}} (\sigma_{\text{color}=\text{red}} \text{Boats} \bowtie \text{Reserves}) \]

- Strategy
  - Find all sailors (sids) with age over 20
  - Find all sailors (sids) who have reserved a red boat
  - Take their set differences

(Q9A) Find the names of sailors who have reserved all boats

\[ \pi_{\text{sname}} (\pi_{\text{sid,bid}} (\text{Reserves}) / \pi_{\text{bid}} (\text{Boats}) \bowtie \text{Sailors}) \]

- Strategy
  - Use division; all = division
  - what divides by what? Solution
(Q9’) Find the names of sailors who have reserved boats with all different colors

\[ \text{Reserves}(\text{sid}, \text{bid}, \text{day}) \]
\[ \text{Sailors}(\text{sid}, \text{sname}, \text{rating}, \text{age}) \]
\[ \text{Boats}(\text{bid}, \text{bname}, \text{color}) \]

- **Strategy**
  - *what divides by what?*
  - \((\text{sid, color}) / (\text{color})\)
- **Solution**

\[ \pi_{\text{sname}} (((\pi_{\text{sid, color}} (\text{Reserves} \bowtie \text{Boats})) / \pi_{\text{color}} (\text{Boats})) \bowtie \text{Sailors}) \]

(Q10) Find the names of sailors who have reserved all boats called “Interlake”

\[ \text{Reserves}(\text{sid}, \text{bid}, \text{day}) \]
\[ \text{Sailors}(\text{sid}, \text{sname}, \text{rating}, \text{age}) \]
\[ \text{Boats}(\text{bid}, \text{bname}, \text{color}) \]

- **Previous solution?**

\[ \pi_{\text{sname}} (((\pi_{\text{sid, bid}} (\text{Reserves})) / \pi_{\text{bid}} (\text{Boats})) \bowtie \text{Sailors}) \]

- **How to modify it?**

\[ \pi_{\text{sname}} (((\pi_{\text{sid, bid}} (\text{Reserves})) / \pi_{\text{bid}} (\sigma_{\text{bname=’Interlake’}} \text{Boats})) \bowtie \text{Sailors}) \]
More Exercises

\[
\text{employee}(\text{person-name, street, city}) \\
\text{works}(\text{person-name, company-name, salary}) \\
\text{company}(\text{company-name, city}) \\
\text{manages}(\text{person-name, manager-name})
\]

(a) Find the names of all employees who work for First Bank Corporation.
(b) Find the names and cities of residence of all employees who work for First Bank Corporation.
(c) Find the names, street address, and cities of residence of all employees who work for First Bank Corporation and earn more than $10,000 per annum.
employee(person-name, street, city)
works(person-name, company-name, salary)
company(company-name, city)
manages(person-name, manager-name)

(d) Find the names of all employees in this database who live in the same city as the company locates for which they work.
(e) Find the names of all employees who live in the same city and on the same street as do their managers.
(f) Find the names of all employees in this database who do not work for the First Bank Corporation.

(g) Find the names of all employees who earn more than every employee of Small Bank Corporation.
(h) Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.
(i) Find the names of employees who work for more than 3 (included) companies.