Announcements

- Assignment 1 is due now!
- Assignment 2 is due next Tuesday.

Outline

- So far, we studied schema design.
- How to manipulate data?
- Relational algebra
  - Elegant theoretical framework
  - Not so elegant in practice – SQL
- Relational operators
- Why decomposition to normal forms works?

Core Relational Algebra

- A small set of operators that allow us to manipulate relations in limited but useful ways.
  1. Union, intersection, and difference: the usual set operators.
     - Relation schemas must be the same.
  2. Selection: Pick certain rows from a relation.
  4. Products and joins: Combine relations in useful ways.
  5. Renaming of relations and their attributes.

Selection

- \( R_1 = \sigma_C(R_2) \)
  - where \( C \) is a condition involving the attributes of relation \( R_2 \).
- Example:
  Relation \( Sells \):
  - Spoon Amstel 4
  - Spoon Guinness 7
  - Whiskey Guinness 7
  - Whiskey Bud 5
  \( SpoonMenu = \sigma_{\text{bar}=\text{Spoon}}(\text{Sells}) \)

Projection

- \( R_1 = \pi_L(R_2) \)
  - where \( L \) is a list of attributes from the schema of \( R_2 \).
- Example
  \( \pi_{\text{beer,price}}(\text{Sells}) \)
  - Amstel 4
  - Guinness 7
  - Bud 5

Notice elimination of duplicate tuples.
Product

- \( R = R_1 \times R_2 \)
  - pairs each tuple \( t_1 \) of \( R_1 \) with each tuple \( t_2 \) of \( R_2 \) and puts in \( R \) a tuple \( t_1t_2 \).
- Theta-Join: \( R = R_1 \bowtie R_2 \)
  - is equivalent to \( R = \sigma_{\theta}(R_1 \times R_2) \).

Example

\[
\begin{array}{|c|c|c|}
\hline
\text{beer} & \text{price} & \text{name} \\
\hline
\text{Spoon} & 4 & \text{Amstel} \\
\text{Spoon} & 7 & \text{Guinness} \\
\text{Whiskey} & 7 & \text{Guinness} \\
\text{Whiskey} & 5 & \text{Bud} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
\text{beer} & \text{price} & \text{name} & \text{addr} \\
\hline
\text{Spoon} & 4 & \text{Spoon} & \text{Wells} \\
\text{Spoon} & 7 & \text{Spoon} & \text{Wells} \\
\text{Whiskey} & 7 & \text{Whiskey} & \text{Rush} \\
\text{Whiskey} & 5 & \text{Whiskey} & \text{Rush} \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|}
\hline
\text{beer} & \text{price} \\
\hline
\text{Rush} & \text{Bud} \\
\text{Rush} & \text{Guinness} \\
\text{Wells} & \text{Guinness} \\
\text{Wells} & \text{Amstel} \\
\hline
\end{array}
\]

Natural Join

- \( R = R_1 \bowtie R_2 \)
  - Equivalent to:
    1. theta-join of \( R_1 \) and \( R_2 \) with the condition that all attributes of the same name be equated.
    2. one column for each pair of equated attributes is projected out.
  - What is the formula?
  - Example:
    - Suppose the attribute name in relation Bars was changed to bar, to match the bar name in Sells.
    - \( \text{BarInfo} = \text{Sells} \bowtie \text{Bars} \)

Natural Join Example

\[
\begin{array}{|c|c|c|}
\hline
\text{beer} & \text{price} & \text{name} & \text{addr} \\
\hline
\text{Spoon} & 4 & \text{Spoon} & \text{Wells} \\
\text{Spoon} & 7 & \text{Spoon} & \text{Wells} \\
\text{Whiskey} & 7 & \text{Whiskey} & \text{Rush} \\
\text{Whiskey} & 5 & \text{Whiskey} & \text{Rush} \\
\hline
\end{array}
\]

Renaming

- \( \rho_{A_1,\ldots,A_n}(R) \) produces a relation identical to \( R \) but named \( S \) and with attributes, in order, named \( A_1,\ldots,A_n \).
- Example:
  - \( \rho_{\text{bar},\text{addr}}(\text{Bars}) = \)
  - \( \rho_{R(\text{bar},\text{addr})}(\text{Bars}) = \)
  - The name of the second relation is \( R \).

Combining Operations

- Any algebra is defined as:
  - basis arguments
  - ways of constructing expressions
- For relational algebra:
  - Arguments = variables standing for relations + finite, constant relations.
  - Expressions constructed by applying one of the operators + parentheses.
  - Query = expression of relational algebra.
Operator Precedence

- The normal way to group operators is:
  1. Unary operators \( \sigma \), \( \pi \), and \( \rho \) have highest precedence.
  2. Next highest are the multiplicative operators, \( \times \), \( \cap \), and \( \times \).
  3. Lowest are the additive operators, \( \cup \), \( \cap \), and \(-\).
- But there is no universal agreement, so we always put parentheses around the argument of a unary operator, and it is a good idea to group all binary operators with parentheses enclosing their arguments.
- Example: Group \( R \cup \sigma S \cap T \) as \( R \cup (\sigma(S) \cap T) \).

Expressions and Schemas

- If \( \cup \), \( \cap \), \(-\) applied, schemas are the same, so the result has the same schema.
- Projection: use the attributes listed in the projection.
- Selection: no change in schema.
- Product \( R \times S \) use attributes of \( R \) and \( S \).
  - But if they share an attribute \( A \), prefix it with the relation name, as \( R.A, S.A \).
- Theta-join: same as product.
- Natural join: use attributes from each relation; common attributes are merged anyway.
- Renaming: whatever it says.

Example 1

- Find the bars that are either on Wells Street or sell Bud for less than $6.
  
  \[ Sells(bar, beer, price) \]
  
  \[ Bars(name, addr) \]

Example 2

- Find the bars that sell two different beers at the same price.
  
  \[ Sells(bar, beer, price) \]

Linear Notation for Expressions

- Invent new names for intermediate relations, and assign them values that are algebraic expressions.
- Renaming of attributes implicit in schema of new relation.

Example

- Find the bars that are either on Wells Street or sell Bud for less than $6.
  
  \[ Sells(bar, beer, price) \]
  
  \[ Bars(name, addr) \]

\[
R1(name) := \pi_{name}(\sigma_{addr = Wells}(Bars)) \\
R2(name) := \pi_{name}(\sigma_{beer=Bud AND price<6}(Sells)) \\
R3(name) := R1 \cup R2
\]
Why Decomposition “Works”? 

- What does it mean to "work"? Why can’t we just tear sets of attributes apart as we like? 
- Answer: the decomposed relations need to represent the same information as the original. 
- We must be able to reconstruct the original from the decomposed relations. 
- Projection and join connect the original and decomposed relations.

Example (1/3)

$$ R = $$

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>beersLiked</th>
<th>manf</th>
<th>favoriteBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>111 E Ohio</td>
<td>Bud</td>
<td>A.B.</td>
<td>Blonde Ale</td>
</tr>
<tr>
<td>Mike</td>
<td>111 E Ohio</td>
<td>Blonde Ale</td>
<td>G.I.</td>
<td>Blonde Ale</td>
</tr>
<tr>
<td>Anna</td>
<td>123 W Grand</td>
<td>BudLite</td>
<td>A.B.</td>
<td>BudLite</td>
</tr>
</tbody>
</table>

- Recall we decomposed this relation as:
  $$ Drinkers1 \rightarrow Drinkers2 $$
  $$ Drinkers3 \rightarrow Drinkers4 $$

Example (2/3)

Project onto Drinkers1(name, addr, favoriteBeer):

<table>
<thead>
<tr>
<th>name</th>
<th>addr</th>
<th>favoriteBeer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>111 E Ohio</td>
<td>Blonde Ale</td>
</tr>
<tr>
<td>Anna</td>
<td>123 W Grand</td>
<td>BudLite</td>
</tr>
</tbody>
</table>

Project onto Drinkers3(beersLiked, manf):

<table>
<thead>
<tr>
<th>beersLiked</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bud</td>
<td>A.B.</td>
</tr>
<tr>
<td>Blonde Ale</td>
<td>G.I.</td>
</tr>
</tbody>
</table>

Example (3/3)

Project onto Drinkers4(name, beersLiked):

<table>
<thead>
<tr>
<th>name</th>
<th>beersLiked</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Bud</td>
</tr>
<tr>
<td>Mike</td>
<td>Blonde Ale</td>
</tr>
<tr>
<td>Anna</td>
<td>BudLite</td>
</tr>
</tbody>
</table>

Reconstruction

- Can we figure out the original relation from the decomposed relations?
- Sometimes, if we natural join the relations.

Example

$$ Drinkers3 \bowtie Drinkers4 = $$

<table>
<thead>
<tr>
<th>name</th>
<th>beersLiked</th>
<th>manf</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mike</td>
<td>Bud</td>
<td>A.B.</td>
</tr>
<tr>
<td>Mike</td>
<td>Blonde Ale</td>
<td>G.I.</td>
</tr>
<tr>
<td>Anna</td>
<td>BudLite</td>
<td>A.B.</td>
</tr>
</tbody>
</table>

- Join of above with Drinkers1 = original R.
Theorem

- Suppose we decompose a relation with schema $XYZ$ into $XY$ and $XZ$ and project the relation for $XYZ$ onto $XY$ and $XZ$. Then $XY \rightarrow XZ$ is guaranteed to reconstruct $XYZ$ if and only if $X \rightarrow Y$ (or equivalently, $X \rightarrow Z$).
- Usually, the MVD is really a FD, $X \rightarrow Y$ or $X \rightarrow Z$.

Implications

- BCNF: When we decompose $XYZ$ into $XY$ and $XZ$, it is because there is a FD $X \rightarrow Y$ or $X \rightarrow Z$ that violates BCNF.
  - Thus, we can always reconstruct $XYZ$ from its projections onto $XY$ and $XZ$.
- 4NF: when we decompose $XYZ$ into $XY$ and $XZ$ it is because there is an MVD $X \rightarrow Y$ or $X \rightarrow Z$ that violates 4NF.
  - Again, we can reconstruct $XYZ$ from its projections onto $XY$ and $XZ$. 