CS 235: Introduction to Databases

Svetlozar Nestorov

Lecture Notes #5

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 allows 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

• \( R_I = \sigma_C(R) \)
  • where \( C \) is a condition involving the attributes of relation \( R \).
• Example:
  Relation \( Sells \):

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoon</td>
<td>Amstel</td>
<td>4</td>
</tr>
<tr>
<td>Spoon</td>
<td>Guinness</td>
<td>7</td>
</tr>
<tr>
<td>Whiskey</td>
<td>Guinness</td>
<td>7</td>
</tr>
<tr>
<td>Whiskey</td>
<td>Bud</td>
<td>5</td>
</tr>
</tbody>
</table>

\( SpoonMenu = \sigma_{bar=\text{Spoon}}(Sells) \)

<table>
<thead>
<tr>
<th>bar</th>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spoon</td>
<td>Amstel</td>
<td>4</td>
</tr>
<tr>
<td>Spoon</td>
<td>Guinness</td>
<td>7</td>
</tr>
</tbody>
</table>

Projection

• \( R_I = \pi_L(R) \)
  • where \( L \) is a list of attributes from the schema of \( R \).
• Example
  \( \pi_{\text{beer,price}}(Sells) \)

<table>
<thead>
<tr>
<th>beer</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amstel</td>
<td>4</td>
</tr>
<tr>
<td>Guinness</td>
<td>7</td>
</tr>
<tr>
<td>Bud</td>
<td>5</td>
</tr>
</tbody>
</table>

• 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 \( t_1 t_2 \).
- Theta-Join: \( R = R_1 \bowtie R_2 \)
  - is equivalent to \( R = \sigma_C(R_1 \times R_2) \).

**Example**

\[
\begin{array}{c|c|c}
\text{beer} & \text{price} & \text{name} \\
\hline
\text{Amstel} & 6 & \text{Rush} \\
\text{Guinness} & 7 & \text{Wells} \\
\text{Bud} & 5 & \text{Rush} \\
\text{Spoon} & 4 & \text{Wells} \\n\end{array}
\]

**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 \( \text{bar} \), to match the \( \text{bar} \) name in \( Sells \).
  - \( \text{BarInfo} = Sells \bowtie Bars \)

**Natural Join Example**

\[
\begin{array}{c|c|c|c|c}
\text{bar} & \text{beer} & \text{price} & \text{name} & \text{addr} \\
\hline
\text{Spoon} & \text{Amstel} & 6 & \text{Rush} & \text{Wells} \\
\text{Spoon} & \text{Guinness} & 7 & \text{Wells} & \text{Rush} \\
\text{Whiskey} & \text{Guinness} & 7 & \text{Wells} & \text{Rush} \\
\text{Whiskey} & \text{Bud} & 5 & \text{Wells} & \text{Rush} \\
\end{array}
\]

**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:
  - \( \rho_{(\text{bar}, \text{addr})}(Bars) = \)

\[
\begin{array}{c|c|c|c|c}
\text{beer} & \text{price} & \text{name} & \text{addr} \\
\hline
\text{Amstel} & 6 & \text{Rush} & \text{Wells} \\
\text{Guinness} & 7 & \text{Wells} & \text{Rush} \\
\text{Bud} & 5 & \text{Wells} & \text{Rush} \\
\end{array}
\]

- 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, $\preceq$, $\preceq\preceq$, 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 $R \cup \sigma S \preceq T$ as $R \cup (\sigma(S) \preceq T)$.

### 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 $6$.
  - $\text{Sells}(\text{bar, beer, price})$
  - $\text{Bars}(\text{name, addr})$

### Example 2
- Find the bars that sell two different beers at the same price.
  - $\text{Sells}(\text{bar, beer, price})$

### 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 $6$.
  - $\text{Sells}(\text{bar, beer, price})$
  - $\text{Bars}(\text{name, addr})$
  - $R1(\text{name}) := \pi_{\text{name}}(\sigma_{\text{addr}=\text{Wells}}(\text{Bars}))$
  - $R2(\text{name}) := \pi_{\text{bar}}(\sigma_{\text{beer}=\text{Bud} \text{ AND } \text{price} < 6}(\text{Sells}))$
  - $R3(\text{name}) := R1 \cup R2$
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

Example (1/3)

\[
R = \begin{array}{ccc}
\text{name} & \text{addr} & \text{beersLiked} & \text{manf} & \text{favoriteBeer} \\
\text{Mike} & 111 \ E \ Ohio & \text{Bud} & A.B. & \text{Blonde Ale} \\
\text{Mike} & 111 \ E \ Ohio & \text{Blonde Ale} & G.I. & \text{Blonde Ale} \\
\text{Anna} & 123 \ W \ Grand & \text{Bud} & A.B. & \text{BudLite} \\
\end{array}
\]

- Recall we decomposed this relation as:

\[
\begin{array}{c}
\text{Drinkers1} \\
\text{Drinkers2} \\
\text{Drinkers3} \\
\text{Drinkers4} \\
\end{array}
\]

Example (2/3)

Project onto \text{Drinkers1}(\text{name, addr, favoriteBeer}): 

\[
\begin{array}{c}
\text{name} & \text{addr} & \text{favoriteBeer} \\
\text{Mike} & 111 \ E \ Ohio & \text{Blonde Ale} \\
\text{Anna} & 123 \ W \ Grand & \text{BudLite} \\
\end{array}
\]

Example (3/3)

Project onto \text{Drinkers3}(\text{beersLiked, manf}): 

\[
\begin{array}{c}
\text{beersLiked} & \text{manf} \\
\text{Bud} & A.B. \\
\text{Blonde Ale} & G.I. \\
\text{BudLite} & A.B. \\
\end{array}
\]

Reconstruction

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

Example

\[
\text{Drinkers3} \bowtie \text{Drinkers4} = \begin{array}{ccc}
\text{name} & \text{beersLiked} & \text{manf} \\
\text{Mike} & \text{Bud} & A.B. \\
\text{Mike} & \text{Blonde Ale} & G.I. \\
\text{Anna} & \text{BudLite} & A.B. \\
\end{array}
\]

- Join of above with \text{Drinkers1} = \text{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 $\bowtie$ 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.