DATABASE TECHNOLOGY - 1DL124

Summer 2007

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-sommar07/
alt. http://www.it.uu.se/edu/course/homepage/dbdesign/st07/

Kjell Orsborn
Uppsala Database Laboratory
Department of Information Technology, Uppsala University,
Uppsala, Sweden
Introduction to Relational Algebra

Elmasri/Navathe ch 6
Padron-McCarthy/Risch ch 10

Kjell Orsborn

Department of Information Technology
Uppsala University, Uppsala, Sweden
Query languages

• Languages where users can express what information to retrieve from the database.

• Categories of query languages:
  – Procedural
  – Non-procedural (declarative)

• Formal (“pure”) languages:
  – Relational algebra
  – Relational calculus
    • Tuple-relational calculus
    • Domain-relational calculus
  – Formal languages form underlying basis of query languages that people use.
Relational algebra

- **Relational algebra** is a procedural language
- Operations in relational algebra takes two or more relations as arguments and return a new relation.
- Relational algebraic operations:
  - Operations from set theory:
    - Union, Intersection, Difference, Cartesian product
  - Operations specifically introduced for the relational data model:
    - Select, Project, Join
- It have been shown that the *select, project, union, difference*, and *cartesian product* operations form a complete set. That is any other relational algebra operation can be expressed in these.
Operations from set theory

• Relations are required to be **union compatible** to be able to take part in the union, intersection and difference operations.

• Two relations $R_1$ and $R_2$ is said to be union-compatible if:

$$R_1 \subseteq D_1 \times D_2 \times \ldots \times D_n \text{ and}$$

$$R_2 \subseteq D_1 \times D_2 \times \ldots \times D_n$$

i.e. if they have the same degree and the same domains.
Union operation

- The **union** of two union-compatible relations $R$ and $S$ is the set of all tuples that either occur in $R$, $S$, or in both.
- Notation: $R \cup S$
- Defined as: $R \cup S = \{ t \mid t \in R \text{ or } t \in S \}$
- For example:

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
\hline
\end{array}
\cup
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 2 \\
b & 3 \\
\hline
\end{array}
= \begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
b & 3 \\
\hline
\end{array}
\]
Difference operation

- The **difference** between two union-compatible sets $R$ and $S$ is the set of all tuples that occur in $R$ but not in $S$.
- Notation: $R - S$
- Defined as: $R - S = \{ t \mid t \in R \text{ and } t \notin S \}$
- For example:

\[
R - S = \begin{array}{cc}
A & B \\
a & 1 \\
a & 2 \\
b & 1 \\
\end{array}
\]

\[
A - B = \begin{array}{cc}
A & B \\
a & 2 \\
b & 3 \\
\end{array}
\]

\[
R - S = \begin{array}{cc}
A & B \\
a & 1 \\
b & 1 \\
\end{array}
\]
Intersection

- The **intersection** of two union-compatible sets $R$ and $S$, is the set of all tuples that occur in both $R$ and $S$.
- Notation: $R \cap S$
- Defined as: $R \cap S = \{t \mid t \in R \text{ and } t \in S\}$
- For example:

\[
\begin{array}{c|c}
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
\end{array}
\bigcap
\begin{array}{c|c}
A & B \\
\hline
a & 2 \\
b & 3 \\
\end{array} =
\begin{array}{c|c}
A & B \\
\hline
a & 2 \\
\end{array}
\]
Cartesian product

- Let $R$ and $S$ be relations with $k_1$ and $k_2$ arities resp. The **cartesian product** of $R$ and $S$ is the set of all possible $k_1 + k_2$ tuples where the first $k_1$ components constitute a tuple in $R$ and the last $k_2$ components a tuple in $S$.
- Notation: $R \times S$
- Defined as: $R \times S = \{t \circ q \mid t \in R \text{ and } q \in S\}$
- Assume that attributes of $r(R)$ and $s(S)$ are disjoint. (i.e. $R \cap S = \emptyset$). If attributes of $r(R)$ and $s(S)$ are not disjoint, then renaming must be used.
Cartesian product example

\[
\begin{array}{c|c}
A & B \\
\hline
a & 1 \\
b & 2 \\
\end{array}
\times
\begin{array}{c|c}
C & D \\
\hline
a & 5 \\
b & 5 \\
b & 6 \\
c & 5 \\
\end{array}
= 
\begin{array}{c|c|c|c}
A & B & C & D \\
\hline
a & 1 & a & 5 \\
a & 1 & b & 5 \\
a & 1 & b & 5 \\
a & 1 & c & 5 \\
b & 2 & a & 5 \\
b & 2 & b & 5 \\
b & 2 & b & 5 \\
b & 2 & c & 5 \\
\end{array}
\]
Selection operation

- The selection operator, \( \sigma \), selects a specific set of tuples from a relation according to a selection condition (or selection predicate) \( P \).
- Notation: \( \sigma_P(R) \)
- Defined as: \( \sigma_P(R) = \{ t \mid t \in R \text{ and } P(t) \} \) (i.e. the set of tuples \( t \) in \( R \) that fulfills the condition \( P \))
- Where \( P \) is a logical expression (*) consisting of terms connected by:
  \( \land \) (and), \( \lor \) (or), \( \neg \) (not)
and each term is one of:
  \(<\text{attribute}> op <\text{attribute}> \text{ or } <\text{constant}>\)
  where \( op \) is one of: \( =, \neq, >, \geq, <, \leq \)

Example: \( \sigma_{\text{SALARY}>30000}(\text{EMPLOYEE}) \)

(*) a formula in propositional calculus
### Selection example

Let $\sigma_{A=B \land D > 5}(R)$ be a selection example.

#### Table $R$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>5</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

#### Table $\sigma_{A=B \land D > 5}(R)$

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
<th>$C$</th>
<th>$D$</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>1</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>b</td>
<td>4</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>
Projection operation

- The **projection** operator, $\Pi$, picks out (or projects) listed columns from a relation and creates a new relation consisting of these columns.

- Notation: $\Pi_{A_1,A_2,...,A_k}(R)$ where $A_1, A_2$ are attribute names and $R$ is a relation name.

- The result is a new relation of $k$ columns.

- Duplicate rows removed from result, since relations are sets.

Example: $\Pi_{LNAME,FNAME,SALARY}(EMPLOYEE)$
Projection example

\[ R = \begin{array}{ccc}
A & B & C \\
an & 1 & 1 \\
ana & 2 & 1 \\
ab & 3 & 1 \\
bb & 4 & 2 \\
\end{array} \]

\[ \Pi_{A,C}(R) = \begin{array}{cc}
A & C \\
an & 1 \\
ana & 1 \\
ab & 1 \\
bb & 2 \\
\end{array} = \begin{array}{cc}
A & C \\
an & 1 \\
ana & 1 \\
bb & 2 \\
\end{array} \]
Join operator

- The **join** operator, $\otimes$ (almost, correct $\Join$), creates a new relation by joining related tuples from two relations.

- Notation: $R \otimes_C S$

  $C$ is the join condition which has the form $A_r \theta A_s$, where $\theta$ is one of $\{=, <, >, \leq, \geq, \neq\}$. Several terms can be connected as $C_1 \land C_2 \land \ldots C_k$.

- A join operation with this kind of general join condition is called “Theta join”.
Example Theta join

\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
6 & 7 & 8 \\
9 & 7 & 8 \\
\hline
\end{array}
\otimes_{A \leq F}
\begin{array}{|c|c|c|}
\hline
D & E & F \\
\hline
2 & 3 & 4 \\
7 & 3 & 5 \\
7 & 8 & 9 \\
\hline
\end{array}
= 
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
1 & 2 & 3 \\
1 & 2 & 3 \\
6 & 7 & 8 \\
9 & 7 & 8 \\
\hline
\end{array}
\otimes_{A \leq F}
\begin{array}{|c|c|c|}
\hline
D & E & F \\
\hline
2 & 3 & 4 \\
7 & 3 & 5 \\
7 & 8 & 9 \\
\hline
\end{array}
\]
Equijoin

• The same as join but it is required that attribute $A_r$ and attribute $A_s$ should have the same value.

• Notation: $R \bowtie C \ S$

$C$ is the join condition which has the form $A_r = A_s$. Several terms can be connected as $C_1 \land C_2 \land \ldots \land C_k$. 
Example Equijoin

\[ R \times_{B=C} S = R \bowtie_{B=C} S \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>4</td>
</tr>
<tr>
<td>a</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>d</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>2</td>
<td>d</td>
<td>e</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>4</td>
<td>4</td>
<td>d</td>
<td>e</td>
<td></td>
</tr>
</tbody>
</table>
Natural join

- **Natural join** is equivalent with the application of join to R and S with the equality condition $A_r = A_s$ (i.e. an equijoin) and then removing the redundant column $A_s$ in the result.

- Notation: $R \ast_{A_r,A_s} S$
  $A_r, A_s$ are attribute pairs that should fulfil the join condition which has the form $A_r = A_s$. Several terms can be connected as $C_1 \land C_2 \land \ldots C_k$. 
Example Natural join

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 2 \\
\hline
a & 4 \\
\hline
\end{array}
\quad \bowtie_{B=C}
\quad
\begin{array}{|c|c|c|}
\hline
C & D & E \\
\hline
2 & d & e \\
\hline
4 & d & e \\
\hline
9 & d & e \\
\hline
\end{array}
\]

\[
\begin{array}{|c|c|c|c|}
\hline
A & B & D & E \\
\hline
a & 2 & d & e \\
\hline
a & 4 & d & e \\
\hline
\end{array}
\]

\[
R \bowtie_{B=C} S
\]
Composition of operations

- Expressions can be built by composing multiple operations
- Example: $\sigma_{A=C}(R \times S)$

\[
\begin{align*}
R \times S & = \begin{pmatrix}
A & B \\
a & 1 \\
b & 2 \\
\end{pmatrix} \times \begin{pmatrix}
C & D \\
a & 5 \\
b & 5 \\
c & 6 \\
\end{pmatrix} = \begin{pmatrix}
A & B & C & D \\
a & 1 & a & 5 \\
a & 1 & b & 5 \\
a & 1 & b & 6 \\
\end{pmatrix} \\
\sigma_{A=C}(R \times S) & = \begin{pmatrix}
A & B & C & D \\
a & 1 & a & 5 \\
\end{pmatrix}
\end{align*}
\]
Assignment operation

- The assignment operation (←) makes it possible to assign the result of an expression to a temporary relation variable.
- Example:
  
  \[
  \text{temp} \leftarrow \sigma_{dno = 5} (\text{EMPLOYEE}) \\
  \text{result} \leftarrow \Pi_{\text{fname}, \text{name}, \text{salary}} (\text{temp})
  \]

- The result to the right of the ← is assigned to the relation variable on the left of the ←.
- The variable may be used in subsequent expressions.
Renaming relations and attribute

- The assignment operation can also be used to rename relations and attributes.

- Example:

\[
\text{NEWEMP} \leftarrow \sigma_{\text{dno} = 5}(\text{EMPLOYEE})
\]

\[
R(\text{FIRSTNAME}, \text{LASTNAME}, \text{SALARY}) \leftarrow \Pi_{\text{fname},\text{lname},\text{salary}} (\text{NEWEMP})
\]
Division operation

• Suited to queries that include the phrase “for all”.
• Let $R$ and $S$ be relations on schemas $R$ and $S$ respectively, where

$$R = (A_1,\ldots,A_m,B_1,\ldots,B_n)$$
$$S = (B_1,\ldots,B_n)$$

• The result of $R \div S$ is a relation on the schema $R - S = (A_1,\ldots,A_m)$

$$R \div S = \{ t \mid t \in \Pi_{R-S}(R) \ \forall u \in S \land tu \in R \}$$
Example Division operation

\[
\frac{R}{S} = \frac{A}{a} = \frac{1}{2}
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td>b</td>
<td>3</td>
</tr>
<tr>
<td>c</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>1</td>
</tr>
<tr>
<td>d</td>
<td>3</td>
</tr>
<tr>
<td>d</td>
<td>4</td>
</tr>
<tr>
<td>d</td>
<td>6</td>
</tr>
<tr>
<td>e</td>
<td>1</td>
</tr>
<tr>
<td>e</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
</tr>
<tr>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
</tr>
<tr>
<td>e</td>
</tr>
</tbody>
</table>
Relation algebra as a query language

- Relational schema: \textit{supplies}(sname, iname, price)
- “What is the names of the suppliers that supply cheese?”
  \(\pi_{\text{sname}}(\sigma_{\text{iname} = 'CHEESE'}(\text{SUPPLIES}))\)
- “What is the name and price of the items that cost less than 5 $ and that are supplied by WALMART”
  \(\pi_{\text{iname}, \text{price}}(\sigma_{\text{sname} = 'WALMART' \land \text{price} < 5}(\text{SUPPLIES}))\)
Additional relational operations

- Outer join and outer union (presented together with SQL)
- Aggregate functions (presented together with SQL)
- Update operations (presented together with SQL)
  - (not part of pure query language)
Aggregation operations

• Presented together with SQL later
• Examples of aggregation operations
  – avg
  – min
  – max
  – sum
  – count
Update operations

• Presented together with SQL later
• Operations for database updates are normally part of the DML
  – insert (of new tuples)
  – update (of attribute values)
  – delete (of tuples)
• Can be expressed by means of the assignment operator