DATABASE DESIGN I - 1DL300

Summer 2008

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-sommar08/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/st08/

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:

$$R_1 \subseteq D_1 \times D_2 \times ... \times D_n \text{ and } \quad R_2 \subseteq D_1 \times D_2 \times ... \times D_n$$

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:**

\[
\begin{array}{c|c}
R & S \\
\hline
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
\end{array}
\bigcup
\begin{array}{c|c}
A & B \\
\hline
a & 2 \\
b & 3 \\
\end{array}
= \begin{array}{c|c}
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
b & 3 \\
\end{array}
\]
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:

\[
\begin{array}{c|c}
A & B \\
--- & --- \\
a & 1 \\
\text{a} & 2 \\
b & 1 \\
\end{array}
\quad - 
\quad
\begin{array}{c|c}
A & B \\
--- & --- \\
a & 2 \\
\text{b} & 3 \\
\end{array}
\quad =
\quad
\begin{array}{c|c}
A & B \\
--- & --- \\
a & 1 \\
b & 1 \\
\end{array}
\]
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:

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

- Let R and S be relations with k₁ and k₂ arities resp. The **cartesian product** of R and S is the set of all possible k₁+k₂ tuples where the first k₁ components constitute a tuple in R and the last k₂ components a tuple in S.
- Notation: R × S
- Defined as: R × S = {t q | t ∈ R and q ∈ S}
- Assume that attributes of r(R) and s(S) are disjoint. (i.e. R ∩ S = ∅). If attributes of r(R) and s(S) are not disjoint, then renaming must be used.
Cartesian product example

\[
\begin{array}{cc}
A & B \\
\text{a} & 1 \\
\text{b} & 2 \\
\end{array}
\times
\begin{array}{cc}
C & D \\
\text{a} & 5 \\
\text{b} & 5 \\
\text{c} & 5 \\
\end{array} =
\begin{array}{cccc}
A & B & C & D \\
\text{a} & 1 & \text{a} & 5 \\
\text{a} & 1 & \text{b} & 5 \\
\text{a} & 1 & \text{c} & 5 \\
\text{b} & 2 & \text{a} & 5 \\
\text{b} & 2 & \text{b} & 5 \\
\text{b} & 2 & \text{c} & 5 \\
\end{array}
\]
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 (a formula in propositional calculus) consisting of terms connected by: $\wedge$ (and), $\lor$ (or), $\neg$ (not)
  and each term is one of:
  - $<\text{attribute}> \ op <\text{attribute}>$ 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

\[
R = \begin{array}{cccc}
A & B & C & D \\
 a & a & 1 & 7 \\
 a & b & 5 & 7 \\
 b & a & 2 & 3 \\
b & b & 4 & 9 \\
\end{array}
\]

\[
\sigma_{A=B \land D > 5}(R) = \begin{array}{cccc}
A & B & C & D \\
 a & a & 1 & 7 \\
 b & b & 4 & 9 \\
\end{array}
\]
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: $\Pi_{A_1, A_2, \ldots, A_k}(R)$ 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: $\Pi_{\text{LNAME}, \text{FNAME}, \text{SALARY}}(\text{EMPLOYEE})$
Projection example

\[
R = \begin{array}{ccc}
A & B & C \\
 a & 1 & 1 \\
a & 2 & 1 \\
b & 3 & 1 \\
b & 4 & 2 \\
\end{array}
\]

\[
\Pi_{A,C}(R) = \begin{array}{cc}
A & C \\
 a & 1 \\
a & 1 \\
b & 1 \\
b & 2 \\
\end{array} = \begin{array}{cc}
A & C \\
a & 1 \\
b & 1 \\
b & 2 \\
\end{array}
\]
Join operator

- The **join** operator, \( \bowtie \) (almost, correct \( \Join \)), creates a new relation by joining related tuples from two relations.
- Notation: \( R \bowtie C \ S \)
  
  \( C \) 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 \( C_1 \land C_2 \land \ldots \land C_k \).
- A join operation with this kind of general join condition is called “Theta join”.


Example Theta join

\[ R \otimes_{A \leq F} S = R \otimes_{A \leq F} S \]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>( \otimes_{A \leq F} )</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td></td>
<td>7</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>8</td>
<td></td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>
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 $C_1 \land C_2 \land \ldots C_k$. 
Example Equijoin

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

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

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

<table>
<thead>
<tr>
<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>
</tr>
<tr>
<td>a</td>
<td>4</td>
<td>4</td>
<td>d</td>
<td>e</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: $R \,*_{A_r, A_s} S$

  $A_r, A_s$ 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

\[
\begin{align*}
R & \quad \otimes \quad B = C \\
 A & \quad B \\
ah & \quad 2 \\
a & \quad 4 \\
\end{align*}
\]

\[
\begin{array}{|c|c|c|}
\hline
C & D & E \\
\hline
2 & d & e \\
4 & d & e \\
9 & d & e \\
\hline
\end{array}
\]

\[
R \quad \otimes_{B = C} \quad S
\]

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

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

<table>
<thead>
<tr>
<th>$R \times S$</th>
<th>$A$</th>
<th>$B$</th>
<th>$\times$</th>
<th>$C$</th>
<th>$D$</th>
<th>$\sigma_{A=C}(R \times S)$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>a</td>
<td>1</td>
<td></td>
<td>a</td>
<td>5</td>
<td>a</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>2</td>
<td></td>
<td>b</td>
<td>5</td>
<td>b</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>2</td>
<td></td>
<td>c</td>
<td>5</td>
<td>c</td>
</tr>
</tbody>
</table>

$$R \times S = \begin{array}{cc}
  a & 1 \\
  b & 2 \\
\end{array} \times \begin{array}{cc}
  a & 5 \\
  b & 5 \\
  c & 6 \\
\end{array} = \begin{array}{cc}
  a & 1 \\
  a & 1 \\
  b & 1 \\
  b & 1 \\
  c & 1 \\
\end{array} \times \begin{array}{cc}
  a & 5 \\
  b & 5 \\
  b & 6 \\
\end{array}$$

$$\sigma_{A=C}(R \times S) = \begin{array}{cccc}
  a & B & C & D \\
  a & 1 & 5 \\
  b & 2 & 5 \\
  b & 5 & 6 \\
\end{array}$$
Assignment operation

• The assignment operation (←) makes it possible to assign the result of an expression to a temporary relation variable.

• Example:

  \[ temp ← \sigma_{dno = 5} (EMPLOYEE) \]
  \[ result ← \Pi_{fname,lname,salary} (temp) \]

• 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:

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

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

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

$$R = (A_1, ..., A_m, B_1, ..., B_n)$$
$$S = (B_1, ..., B_n)$$

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

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

\[
\begin{array}{c|c}
R & S \\
\hline
A & B \\
\hline
a & 1 \\
ad & 2 \\
abe & 3 \\
abc & 1 \\
abd & 1 \\
dd & 1 \\
dd & 3 \\
dd & 4 \\
ddee & 6 \\
ddee & 1 \\
ddee & 2 \\
\end{array}
\]

\[
R \div S = A
\]

\[
\begin{array}{c|c}
B & A \\
\hline
1 & a \\
2 & e \\
\end{array}
\]
Relation algebra as a query language

• Relational schema: \textit{supplies}((\textit{sname}, \textit{iname}, \textit{price})

• “What is the names of the suppliers that supply cheese?”
  \[ \pi_{\textit{sname}}(\sigma_{\textit{iname}=\text{CHEESE}}(SUPPLIES)) \]

• “What is the name and price of the items that cost less than 5 $ and that are supplied by WALMART”
  \[ \pi_{\textit{iname},\textit{price}}(\sigma_{\textit{sname}=\text{WALMART} \land \textit{price} < 5}(SUPPLIES)) \]
Additional relational operations

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