DATABASE TECHNOLOGY - 1MB025
(also 1DL029, 1DL300+1DL400)

Spring 2008

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-vt2008/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/vt08/

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:

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

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:

$$
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
\hline
\end{array} \cup 
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 2 \\
b & 3 \\
\hline
\end{array} =
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
b & 3 \\
\hline
\end{array}
$$
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: $R - S = \{t \mid t \in R \text{ and } t \not\in S\}$
- For example:

\begin{align*}
R & \quad S \\
\{a, b\} & \quad \{a, b, 1\} \\
\end{align*}

\begin{align*}
\begin{array}{ccc}
A & B \\
a & 1 \\
a & 2 \\
b & 1 \\
\end{array}
\begin{array}{ccc}
A & B \\
a & 2 \\
b & 3 \\
\end{array}
\begin{array}{ccc}
A & B \\
a & 1 \\
b & 1 \\
\end{array}
\end{align*}
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:

$$
\begin{align*}
R &= \{ a, a, b \} \\
S &= \{ a, b \}
\end{align*}
$$

$$
\begin{array}{c|c}
A & B \\
\hline
a & 1 \\
a & 2 \\
b & 1 \\
\end{array}
\quad \cap 
\quad
\begin{array}{c|c}
A & B \\
\hline
a & 2 \\
b & 3 \\
\end{array}
= 
\begin{array}{c|c}
A & B \\
\hline
a & 2 \\
\end{array}
$$
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 $k_1 + k_2$ 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: $R \times S = \{t \circ q | t \in R \text{ and } q \in S\}$
- 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

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

  \(<\text{attribute}> \ op \ <\text{attribute}> \) or \(<\text{constant}> \)

  where \( op \) is one of: \( =, \neq, >, \geq, <, \leq \)

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

\(\text{\textit{\(*)\) a formula in propositional calculus\)}}\)
Selection example

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

\[ \sigma_{A=B \land D > 5} (R) = \begin{array}{c|c|c|c}
  A & B & C & D \\
  a & a & 1 & 7 \\
  b & b & 4 & 9 \\
\end{array} \]
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: \( \Pi_{A_1,A_2,...,A_k}(R) \) 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 = \[
\begin{array}{ccc}
 A & B & C \\
 a & 1 & 1 \\
a & 2 & 1 \\
b & 3 & 1 \\
b & 4 & 2 \\
\end{array}
\]

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

Let's consider a theta join, denoted by $\Theta_{A \leq F}$, between two relations $R$ and $S$.

<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>6</td>
<td>7</td>
<td>8</td>
<td>7</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>8</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>

The join is performed on the condition $A \leq F$. The result is:

$$R \Theta_{A \leq F} S$$
Equijoin

- The same as join but it is required that attribute $A_r$ and attribute $A_s$ should have the same value.
- Notation: $R \otimes C \ S$
  $C$ 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

Let's consider two relations \( R \) and \( S \):

\[
R = \begin{bmatrix}
A & B \\
a & 2 \\
a & 4
\end{bmatrix}
\quad \quad \quad
S = \begin{bmatrix}
C & D & E \\
2 & d & e \\
4 & d & e \\
9 & d & e
\end{bmatrix}
\]

We perform an equijoin on the columns \( B = C \):

\[
R \boxdot_{B=C} S = \begin{bmatrix}
A & B & C & D & E \\
a & 2 & 2 & d & e \\
a & 4 & 2 & d & e
\end{bmatrix}
\]

The resulting relation is:

\[
R \boxdot_{B=C} S = \begin{bmatrix}
A & B & C & D & E \\
a & 2 & 2 & d & e \\
a & 4 & 2 & d & e
\end{bmatrix}
\]
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: $R \star_{A_r, A_s} S$

$A_r, A_s$ 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

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

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

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

$$R \times S = \begin{array}{c|c}
A & B \\
\hline
a & 1 \\
b & 2
\end{array} \times \begin{array}{c|c}
C & D \\
\hline
a & 5 \\
b & 5 \\
b & 6 \\
c & 5
\end{array} = \begin{array}{c|c|c|c}
A & B & C & D \\
\hline
a & 1 & a & 5 \\
a & 1 & b & 5 \\
a & 1 & b & 6 \\
a & 1 & c & 5 \\
b & 2 & a & 5 \\
b & 2 & b & 5 \\
b & 2 & c & 5 \\
b & 2 & 6 & 5
\end{array}$$
Assignment operation

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

• Example:

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

• 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:

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

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

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

\[
\begin{align*}
R & = (A_1, \ldots, A_m, B_1, \ldots, B_n) \\
S & = (B_1, \ldots, B_n)
\end{align*}
\]

• 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

\[
\begin{array}{c|c|c}
R & S & R \div S \\
\hline
A & B & \frac{B}{A} \\
\hline
a & 1 & 1 \\
a & 2 & 2 \\
a & 3 & 3 \\
\hline
b & 1 & 1 \\
c & 1 & 1 \\
d & 3 & 3 \\
d & 4 & 4 \\
d & 6 & 6 \\
de & 1 & 1 \\
e & 2 & 2 \\
\end{array}
\]
Relation algebra as a query language

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