DATABASE TECHNOLOGY - 1MB025

Fall 2004

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-ht2004/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/ht04/

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

Elmasri/Navathe ch 6

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:

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

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:

```
  R   S
  a 1  a 2  a 2
  a 2  b 1  b 3
```

$$R \cup S = \begin{array}{cc}
  a & 1 \\
  a & 2 \\
  b & 1 \\
  b & 3 \\
\end{array}$$
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:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>R</td>
<td>a</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td>a</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>b</td>
<td>3</td>
</tr>
</tbody>
</table>

$R - S = A B$

$= \begin{array}{cc}
  A & B \\
  a & 1 \\
  b & 1
\end{array}$
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:

$$
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 1 \\
\hline
a & 2 \\
b & 1 \\
\hline
\end{array}
\cap
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 2 \\
b & 3 \\
\hline
\end{array}
= \begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 2 \\
\hline
\end{array}
$$
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: \( R \times S = \{ t \cdot q \mid t \in R \text{ and } q \in S \} \)

• 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|}
\hline
A & B \\
\hline
a & 1 \\
b & 2 \\
\hline
\end{array}
\times
\begin{array}{|c|c|}
\hline
C & D \\
\hline
a & 5 \\
b & 5 \\
b & 6 \\
c & 5 \\
\hline
\end{array} =
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 1 \\
a & 1 \\
a & 1 \\
b & 2 \\
b & 2 \\
\hline
C & D \\
\hline
1 & a \\
a & b \\
a & b \\
b & a \\
b & b \\
\hline
\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:
  $<$attribute$>$ $op$ $<$attribute$>$ or $<$constant$>$
  where $op$ is one of: $=$, $\neq$, $>$, $\geq$, $<$, $\leq$

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

(*) a formula in propositional calculus
Selection example

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

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

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

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

- The **join** operator, $\Join$ (almost), creates a new relation by joining related tuples from two relations.
- Notation: $R \Join C S$
  
  $C$ 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

\[ R \bowtie_{A \leq D} S = R \bowtie_{A \leq D} S \]

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>7</td>
<td>8</td>
</tr>
</tbody>
</table>

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
</tbody>
</table>

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>9</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: $R \otimes C \ S$

  $C$ is the join condition which has the form $A_r = A_s$.

  Several terms can be connected as $C_1 \land C_2 \land ... C_k$. 
Example Equijoin

\[
\begin{array}{cc}
R & S
\end{array}
\]

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

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

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 $C_1 \land C_2 \land ... C_k$. 
Example Natural join

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

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

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>a</td>
<td>2</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<td>a</td>
<td>a</td>
<td>4</td>
<td>d</td>
<td>e</td>
</tr>
</tbody>
</table>
Composition of operations

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

\[
\begin{align*}
R \times S & = \\
A & | B \\
a & | 1 \\
b & | 2 \\
C & | D \\
a & | 5 \\
b & | 5 \\
b & | 6 \\
c & | 5 \\
\end{align*}
\]

\[
\sigma_{A=C} (R \times S) = \\
A & | B & C & D \\
a & | 1 & a & 5 \\
b & | 2 & b & 5 \\
b & | 2 & b & 6 \\
\]

\[
A & | B \\
a & | 1 \\
b & | 5 \\
b & | 5 \\
c & | 6 \\
\]

\[
A & | B & C & D \\
a & | 1 & b & 5 \\
b & | 2 & a & 5 \\
c & | 5 & c & 5 \\
\]

\[
A & | B & C & D \\
a & | 1 & b & 6 \\
b & | 2 & a & 5 \\
c & | 5 & c & 5 \\
\]

\[
A & | B & C & D \\
a & | 1 & b & 5 \\
b & | 2 & a & 5 \\
\]

\[
A & | B & C & D \\
a & | 1 & b & 6 \\
b & | 2 & a & 5 \\
\]
Assignment operation

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

- Example:
  
  $\text{temp} ← \sigma_{dno = 5}(\text{EMPLOYEE})$

  $\text{result} ← \Pi_{\text{fname}, \text{lname}, \text{salary}}(\text{temp})$

- The result to the right of the ← is assigned to the relation variable on the left of the ←.

- The variable may use variable in subsequent expressions.
Renaming relations and attribute

- The assignment operation can also be used to rename relations and attributes.
- Example:
  \[ NEWEMP \leftarrow \sigma_{dno = 5}(EMPLOYEE) \]
  \[ R(FIRSTNAME, LASTNAME, SALARY) \leftarrow \Pi_{fname, lname, salary}(NEWEMP) \]
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 schema $R - S = (A_1,\ldots,A_m)$
  \[ R \div S = \{ t \mid t \in \Pi_{R-S}(R) \cdot \ " u \in S (tu \in R) \} \]
Example Division operation

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

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