DATABASE TECHNOLOGY - 1MB025

Fall 2005

An introductory course on database systems

http://user.it.uu.se/~udbl/dbt-ht2005/
alt. http://www.it.uu.se/edu/course/homepage/dbastekn/ht05/

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:

$$R_1 \subseteq D_1 \times D_2 \times \ldots \times D_n$$

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**:

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

$R \cup S = \{ a, a, b \}$

<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>b</td>
<td>b</td>
<td>3</td>
</tr>
</tbody>
</table>

$= \{ a, a, b \}$
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:
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: $R \cap S = \{t | t \in R \text{ and } t \in S\}$
- For example:
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

\[
\begin{array}{c|c}
A & B \\
\hline
a & 1 \\
b & 2 \\
\end{array}
\times
\begin{array}{c|c}
C & D \\
\hline
a & 5 \\
b & 5 \\
b & 6 \\
c & 5 \\
\end{array}
= 
\begin{array}{c|c|c|c}
A & B & C & D \\
\hline
a & 1 & a & 5 \\
a & 1 & b & 5 \\
a & 1 & b & 6 \\
a & 1 & c & 5 \\
b & 2 & a & 5 \\
b & 2 & b & 5 \\
b & 2 & b & 6 \\
b & 2 & c & 5 \\
\end{array}
\]
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: $\sigma_P(R) = \{ t | t \in R \text{ and } P(t) \}$ (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: $\Pi_{LNAME, FNAME, SALARY}(EMPLOYEE)$
Projection example

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

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

= \begin{bmatrix}
A & C \\
\text{a} & 1 \\
\text{b} & 1 \\
\text{b} & 2 \\
\end{bmatrix}
Join operator

• The join operator, ⊗ (almost, correct □), creates a new relation by joining related tuples from two relations.

• Notation: R ⊗ 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 \( C_1 \land C_2 \land ... C_k \).

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

\[
R \; \bigotimes_{A \leq D} \; S
\]

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</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>9</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th>C</th>
<th>D</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>

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>B</th>
<th>C</th>
<th>D</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

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

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
a & 2 \\
\hline
a & 4 \\
\hline
\end{array}
\odot_{B=C}

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

= \[
\begin{array}{|c|c|c|c|c|}
\hline
A & B & C & D & E \\
\hline
a & 2 & 2 & d & e \\
\hline
a & 4 & 4 & d & e \\
\hline
\end{array}
\]
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: $R \ast A_r, A_s \ S$
  
  $A_r, A_s$ are attribute pairs that should fulfil the join condition which has the form $A_r = A_s$. Several terms can be connected as $C_1 \land C_2 \land ... C_k$. 
Example Natural join

<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>

\( \otimes_{B=C} \)

<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>

\( R \times_{B=C} S \)

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>2</td>
<td>d</td>
<td>e</td>
</tr>
<tr>
<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: $\sigma_{A=C}(R \times S)$

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

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

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

• Example:

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

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

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

- The assignment operation can also be used to rename relations and attributes.
- Example:
  \[
  \text{NEWEMP} \leftarrow \sigma_{\text{dno} = 5}(\text{EMPLOYEE}) \\
  \Pi_{\text{fname}, \text{lname}, \text{salary}}(\text{NEWEMP}) \\
  \]

\[
\text{R(firstname, lastname, salary)} \leftarrow \\
\text{NEWEMP}
\]
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 schema
  $R \cdot S = (A_1,\ldots,A_m)$
  $R \div S = \{ t \mid t \in \Pi_{R-S}(R) \ \forall u \in S \land tu \in R \}$
Example Division operation

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

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