Relational algebra

A model of how to work with the database

What to do with the database
– Fetch data
– Maintain data

Relational Algebra specifies the operations that may be used in computing new relations from existing ones

Query:
– Starting point: the current state of the database consists of a collection of relations
– produces a new relation as its result.
– is an expression that specifies how the result relation is produced from the starting relations by applying the operations of the relational algebra

Maintenance:
– Relational algebra considers only how to construct new relations of existing ones
– We may consider maintenance as replacing an existing relation with the result of a query
Example:
• Starting point: relation CAR
• Query: Find out all cars but not the one with register number ABC-123. \( = \) CAR1
• Replace the relation CAR with CAR1

Operations to build new relations from existing ones

Basis in mathematical Set Theory

The kernel operations of the Set Theory
– union, difference, (cross) product, intersection

Native operations
– projection, select, joins

Union

• Union builds up a relation that contains all the tuples of the operand relations.
• \( R \cup S = \{ t \mid t \in R \lor t \in S \} \).
• \( R \) and \( S \) are relations and \( t \) is a tuple of either \( R \) or \( S \) or both. A tuple in both \( R \) and \( S \) is included in the result only once.

Difference \( R-S \) extracts from relation \( R \) those tuples that do not belong to \( S \):
\[ R - S = \{ t \mid t \in R \land t \notin S \}. \]
Both Union and Difference presuppose that the operands are **compatible**:
- Same number of attributes
- Corresponding columns share the same domains
- Names of the corresponding columns need not be the same
- The first operand determines the column names for the result

**Product** $R \times S$ produces a relation where each tuple of $R$ is concatenated with each tuple of $S$

If relations have columns with the same name:
- Attach schema name as a specifier
  - $R(A,B,C) \times S(B,C,D) => RxS(A, R.B, R.C, S.B, S.C, D)$

**Projection** extracts from a relation the value combinations appearing in listed columns
- $\pi_{A_1, \ldots, A_n}(R) = \{(a_1, \ldots, a_n) | x \in R, \forall i=1..n: a_i=x.A_i\}$
  - $A_1, \ldots, A_n$ are attributes (column names)
  - $a_1, \ldots, a_n$ are values
  - $x$ is a tuple (row)
  - $x.A$ denotes the value of attribute $A$ in tuple $x$

Although the same value combination $a_1, \ldots, a_n$ would be included in many tuples of the starting relation it is included only once in the result = elimination of duplicates

**Selection** operation extracts from a relation the rows that fulfill a given condition
- $\sigma_{\text{condition}}(R) = \{ x | x \in R \text{ and condition is true when the attribute names in it are substituted by the values of those attributes in tuple } x \}$

Operand in a condition may be constraints or attributes. Standard comparison operations may be used $=, \neq, <, >, \leq, \geq$. 
Relational algebra - selection

\[ \sigma_{A=1}(R) \]

\[ \sigma_{A=5}(R) \]

\[ \sigma_{3>2}(R) \]

Empty relation

Relational algebra - assignment, renaming

Assignment renames the relation

\[ S(A,B,...,N) := \text{expression.} \]

The result of the expressions must be compatible with the schema on the left

Assignment is an auxiliary operation

\[ \text{StudentName(Name)} := \pi_{\text{lastName}}(\text{Student}) \]

Relational algebra - intersection

Intersection is a set theoretic operation that picks up the common rows in two relations, each row, however, only once

\[ R \cap S = R - (R - S) = S - (S - R) \]

Relational algebra - logical expressions in conditions

It is possible to use in complex conditions using similar logical connectives that are used in programming languages. A selection with a complex condition may be reduced to selections with simple conditions

\[ \sigma_{c_1 \text{ or } c_2}(R) = \sigma_{c_1}(R) \cup \sigma_{c_2}(R) \]

\[ \sigma_{c_1 \text{ and } c_2}(R) = \sigma_{c_1}(R) \cap \sigma_{c_2}(R) \]

\[ \sigma_{\text{not } c_1}(R) = R - \sigma_{c_1}(R) \]

Relational algebra - join

Concatenation of rows that satisfy a given condition

Join combines selection and product

\[ R \mid \text{join_condition} \ S = \sigma_{\text{join_condition}}(R \times S) \]

Most common join is to join rows with rows that refer to them

– Join condition is this case is \( R.K = S.FK \),
– where \( K \) is the key of \( R \) and \( FK \) is a foreign key in \( S \) that contains a reference to \( R \)

Join

Condition: \( R.A = S.D \)

First product
Join

\[ R \bowtie S \]

Condition: \( R.A = S.D \)

Then select

Natural join

- \( R \bowtie S \)
- Join condition need not be expressed, but is based on the equality of values of similarly named columns.
  - The same column name in both relations.
  - Similarly named columns are included in the result only once.

Outer join

- Outer join is a combination of union and join. It includes in the result also such rows that in normal join would not be included because they do not have any matching pair to satisfy the join condition (below left outer join)
  - \( R \bowtie_{\text{left}} S = (R \times S) \cup (\pi_{\text{att}(R)} (R \times S)) \times S \)
  - \text{att}(R) lists the attributes of R and \( \pi_{\text{att}(S)} S \) is relation with the schema of S and a single tuple each value of which is a null value.
Ulkoliitos

- Car \( \sqsubset \sqsupset \) Car.Reg_no=Ownership.Reg_no Ownership

  - All cars are included but if no ownership is defined for some car the Ownership attributes in the concatenated tuple are null values.

### Outer join

<table>
<thead>
<tr>
<th>Auto</th>
<th>Rakonto</th>
<th>Omistaja</th>
<th>Reg_no</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC-123</td>
<td>ABC-123</td>
<td>Liisa</td>
<td>ABC-123</td>
</tr>
<tr>
<td>ABC-123</td>
<td>ABC-123</td>
<td>Kalle</td>
<td>ABC-123</td>
</tr>
<tr>
<td>GHI-789</td>
<td>GHI-789</td>
<td>Pekka</td>
<td>GHI-789</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>R-Car</th>
<th>O-Car</th>
<th>O-Masko</th>
</tr>
</thead>
<tbody>
<tr>
<td>ABC-123</td>
<td>ABC-123</td>
<td>Lisa</td>
</tr>
<tr>
<td>ABC-123</td>
<td>ABC-123</td>
<td>Kalle</td>
</tr>
<tr>
<td>GHI-789</td>
<td>GHI-789</td>
<td>Pekka</td>
</tr>
<tr>
<td>DEF-456</td>
<td>NULL</td>
<td>NULL</td>
</tr>
</tbody>
</table>