1 Review of Relational Model

Relational Model. Relations are tables representing information. Columns are headed by attributes; each attribute has an associated domain (also: data type). Rows are called tuples, and a tuple has one component for each attribute of the relation.

Example: a relation about movies

<table>
<thead>
<tr>
<th>Title</th>
<th>Year</th>
<th>Length</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harry Porter</td>
<td>2001</td>
<td>180</td>
<td>Color</td>
</tr>
<tr>
<td>Gone with the wind</td>
<td>1938</td>
<td>125</td>
<td>Black and White</td>
</tr>
<tr>
<td>Snow White</td>
<td>1950</td>
<td>90</td>
<td>Color</td>
</tr>
</tbody>
</table>

The Movie Relation

Relation schema: The name of the relation and the set of attributes for the relation.

Example:

\[
\text{Movie}(\text{Title}, \text{Year}, \text{Length}, \text{Type})
\]

is the relation schema of a relation named \text{Movie} (table above) and the set of attributes

\[
\{\text{Title}, \text{Year}, \text{Length}, \text{Type}\}.
\]

Tuples: Each row of the relation is a tuple. In the above

\[(\text{HarryPorter}, 2001, 180, \text{Color})\]

is a tuple.

Relation Instances: the set of tuples belonging to the relation.

\(1\text{Data are invented to illustrate the concepts. They might or might not be accurate.}\)
2 Relational Algebra

Relational Algebra consists of

1. Atomic operands that are relations or constants;

2. Operations that are divided into the following classes:
   (a) Set operations: union, intersection, and set difference
   (b) Operations that remove part of the relation: projection or selection
   (c) Operations that combine part of the relation: cartesian product or joint
   (d) Renaming operations: rename the schema of the relation

3 Set operations

Union (∪): $R ∪ S$ represents a new relation whose tuples are either tuples belonging to $R$, to $S$, or both.

Set difference (\): $R \setminus S$ represents a new relation whose tuples are tuples belonging to $R$ but not to $S$.

Intersection (∩): $R ∩ S$ represents a new relation whose tuples are tuples belonging to $R$ and $S$.

Note: Set operations only work on relations with the same schema.

Example 1 Let $R$ be the following relation:

<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>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

and $S$ be the following relation:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

Then: $R ∪ S$ is
NOTE: If the schema of S changes, say $S(A_1, B_1, C_1)$ then all of the three operations ($\cup$, $\cap$, $\setminus$) will yield the empty relation.

### 4 Operations that remove part of the relation

**Projection ($\pi$)**  This operation removes some columns from a relation. We write $\pi_{A_1,\ldots,A_k}(R)$ and the result is a new relation that has only the columns $A_1, \ldots, A_k$ of $R$ and whose schema is the set of attributes $\{A_1, \ldots, A_k\}$ of the relation $R$.

For the relation $R$ in Example 1,

$$\pi_{A,B}(R)$$

is the relation:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
**Selection** \((\sigma)\) The operation selects some tuples from the current relation and defines a new relation. \(\sigma_C(R)\) represents a new relation that has tuples belonging to \(R\) and each of them satisfying the conditional expression (or condition) \(C\).

\(C\) is a conditional expression constructed from attributes of \(R\) or constants. For example,

\[
A < B + C
\]

would be a valid conditional expression with respect to the relation \(R\) in Example 1.

For the relation \(R\) in Example 1, \(\sigma_{A < B + C}(R)\) is the following relation:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

**5 Operations that combine relations**

**Cartesian Product** \((\times)\) Given two relations \(R\) and \(S\). The Cartesian product \(R \times S\) is the set of pairs that can be formed by choosing the first element of the pair to be any element of \(R\) and the second an element of \(S\).

**Example 2** Let \(R\) be the following relation:

<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>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

and \(S\) be the following relation:

<table>
<thead>
<tr>
<th>A</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>

Then:

\(R \times S\) is

<table>
<thead>
<tr>
<th>A.R</th>
<th>B</th>
<th>C</th>
<th>A.S</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
<td>2</td>
<td>1</td>
</tr>
</tbody>
</table>
Natural Joins ($\bowtie$) Combine two relations but using only tuples that are matching in some way. Natural join considers only those pairs that have identical values on whatever attributes the two relations share. $R \bowtie S$ is a new relation whose schema contains the attributes in $R$ and those attributes in $S$ that are not in $R$. It has the tuples formed by pairing a tuple in $R$ with a tuple in $S$ provided that both have the same values on attributes shared by $R$ and $S$.

Example 3 Let $R$ be the following relation:

<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>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

and $S$ be the following relation:

<table>
<thead>
<tr>
<th>B</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
</tr>
</tbody>
</table>

Then:

$R \bowtie S$ is

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>

The first tuple of $R \bowtie S$ is combined from the 1st tuple of $R$ and the 1st tuple of $S$. The second tuple of $R \bowtie S$ is combined from the 3rd tuple of $R$ and the 1st tuple of $S$. For the second tuple of $S$, the value of $B$ is 3 and none of the tuples in $R$ has $B = 3$. So, no new tuple for $R \bowtie S$ can be formed using the second tuple of $S$.

Theta-join This is a combination of Cartesian product and selection. $R \bowtie_C S$ is defined by

$$R \bowtie_C S = \sigma_C(R \times S).$$

6 Renaming Operation

Renaming ($\rho$): $\rho_{S(A_1, \ldots, A_k)}(R)$ creates a new relation, named $S$, and attributes $\{A_1, \ldots, A_k\}$ with all the tuples of $R$.

Example 4 Let $R$ be the following relation:
Then, $\rho_{T(M,N,P)}(R)$ is a new relation $T$ with the following instance

<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>2</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

NOTE: $\rho_{S}(R)$ only changes the name of the relation.

7 Combining Operations to Form Queries

The operations of relational algebra allows us to form expressions of arbitrary complexity by applying operations to either a given relation or to the relations that are the result of applying operations to relations.

Example like the theta-join operation, or something like $\pi_{A,B}(\sigma_{C > D}(R \times S))$ etc.

A relation algebraal expression can be represented as an expression tree similar to a evaluation tree of an arithmetic expression in imperative language such as C, PASCAL etc. For example, an expression tree for the query

$$\pi_{\text{title}, \text{year}}(\sigma_{\text{length} \geq 100}(\text{Movie}) \cap \sigma_{\text{studioName} = 'Fox'}(\text{Movie}))$$

is
8 Relational Operations on Bags

The aforementioned relational operations are ‘classical’ operations which operated on sets. So, no duplication of tuples is allowed. The same operations can be defined for bags as well. The operations are essential the same, the only difference is that the result can contain duplication of tuples. For example

**Example 5** Let $R$ be the following relation:

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

and $S$ be the following relation:

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

Then:

$R \cup S$ is

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

$R \cap S$ is

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

and

$R \setminus S$ is

\[
\begin{array}{ccc}
A & B & C \\
1 & 2 & 3 \\
3 & 2 & 2 \\
\end{array}
\]
Formal definitions:

\[ R \cup S = \{ t \mid t \text{ appears } p + q \text{ times, where } t \text{ appears } p \text{ and } q \text{ times in } R \text{ and } S, \text{ respectively} \} . \]

\[ R \cap S = \{ t \mid t \text{ appears } \min(p,q) \text{ times, where } t \text{ appears } p \text{ and } q \text{ times in } R \text{ and } S, \text{ respectively} \} . \]

\[ R \setminus S = \{ t \mid t \text{ appears } \max(0,p-q) \text{ times, where } t \text{ appears } p \text{ and } q \text{ times in } R \text{ and } S, \text{ respectively} \} . \]

Projection, selection, join – are defined as previously, with the exception that the results can contain duplication.

For the relation \( R \) in Example 5, \( \pi_{A,B}(R) \) is the bag \( \{(1,2),(1,2),(3,2),(4,1)\} \).

The selection \( \sigma_{A<A}(R) \) is the bag of two tuples \( \{(1,2,3),(1,2,3)\} \).

The join between \( R \) and \( \rho_{T(B,D,E)}(S) \), i.e., \( R \bowtie \rho_{T(B,D,E)}(S) \) would contain four tuples of which \( (1,2,3,3,4) \) occurs two times and \( (1,2,3,1,1) \) occurs two times.

### 9 Extended Operators of Relational Algebra

The extended relational algebra contains several extended operators (that allow us to express several queries that are impossible using the classical operations):

1. **Duplicate-elimination**, denoted by \( \delta \), \( \delta(R) \) is a relation that is obtained from \( R \) by removing duplications;

2. **Aggregation operator**. This includes: *sum*, *averages*, *min*, *max*, *count*. They are not operators on relations but are used by grouping operator; apply to attributes of a relation and produce a number according to the operator;

3. **Grouping operator**: partitioning the relations into “groups”; then applying the aggregation operators to compute certain number over these groups. The operator \( \gamma \) combines the effect of groups and aggregation.

4. **Sorting operator**: denoted by \( \tau \), turns a relation into a list, sorted according to one or more attributes;

5. **Extended projection**: additional power to the operator \( \pi \) (creating new column from the old ones);

6. **Outerjoin**: avoid losing dangling tuples

The definition for each operator is given next.

For the relation \( R \) from Example 5, \( \delta(R) \) is the relation

<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>3</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
The aggregation operators:

1. $SUM$ produces the sum of a column with numerical value;
2. $AVG$ produces the averages of a column with numerical value
3. $MIN$ and $MAX$, applied to a column with numerical value, produces the smallest or the largest value, respectively. When applied to a column with character-string values, they produce the lexicographically first or last value, respectively.
4. $COUNT$ produces the number of values in a column;

Consider again the relation $R$ from Example 5:
$SUM(A)$ produces the number 9.
$MIN(B)$ produces the number 1.
$MAC(C)$ produces the number 3.
$AVG(A)$ produces the number 2.25.
$COUNT(A)$ (or $COUNT(B)$, $COUNT(C)$) produces the number 4.

The grouping operator: specified by $\gamma_L(R)$ where

1. $R$ is a relation we want to group
2. $L$ is a list of elements, each of which is either
   (a) an attribute of $R$ – called grouping attribute – this attribute is one of the attribute by which $R$ will be grouped;
   (b) a formula of the form $OPR(\text{att}) \rightarrow \text{attName}$ where $OPR$ is one of the aggregation operator applied to an attribute $\text{att}$ of the relation and $\text{attName}$ is the name provided for the attribute corrsponding to this aggregation;

For example, given the relation $StarsIn(title, year, starName)$, the following is a group expression:

$$\gamma_{\text{starName}, MIN(\text{year}) \rightarrow \text{minYear}, COUNT(\text{title}) \rightarrow \text{ctTitle}}(StarsIn).$$ (9.1)

The result of a grouping expression $\gamma_L(R)$ is produced in the following way:

1. partition the tuples of $R$ into groups – each group consists of all tuples having one particular assignment of values to the grouping attributes in the list $L$;
2. for each group, produce one tuple consisting of
   (a) the grouping attributes’s values for that group and
   (b) the aggregations, over all tuples of that group, for the aggregated attributes on list $L$.  

For the expression (9.1) we need to:

(a) partition the relations into groups of tuples whose starName is the same
(b) for each group, produce one tuple with the starName, the earliest year she/he plays in a movie, and the number of the movies she/he plays in.

**Extended Projection Operator.** Denoted by $\pi_L(R)$. Originally, $L$ is a set of attributes; now $L$ can contain also formula of the form $expAtr \rightarrow attName$ where $expAtr$ is an expression over the attributes of $R$ and $attName$ is an name for the newly created attribute.

Consider the relation $R$ in Example 5, the result of the selection $\pi_{A,B+C\rightarrow D}$ is the relation

$$
\begin{array}{|c|c|}
\hline
A & D \\
\hline
1 & 5 \\
1 & 5 \\
3 & 4 \\
4 & 4 \\
\hline
\end{array}
$$

**The Sorting Operator.** Denoted by $\tau_L(R)$ where $L$ is a list of attributes. This will create a new relation in which tuples of $R$ are sorted by the list $L$. If $L = A_1, \ldots, A_n$ then the sorting will begin with $A_1$, then $A_2$, then $A_3$, ldots, and finally $A_n$. For the relation $R$ in Example 5, the expression $\tau_{B,C,A}(R)$ will give the following relation:

$$
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
4 & 1 & 3 \\
3 & 2 & 2 \\
1 & 2 & 3 \\
1 & 2 & 3 \\
\hline
\end{array}
$$

**Outerjoin.** Denoted by $R \bowtie S$. Introduced null values (denoted by $\perp$) whenever the tuple does not have a match from the other relation. Let $R$ be the following relation:

$$
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 8 & 9 \\
\hline
\end{array}
$$

and $S$ be the following relation:

$$
\begin{array}{|c|c|c|}
\hline
B & C & D \\
\hline
2 & 3 & 10 \\
2 & 3 & 11 \\
6 & 7 & 12 \\
\hline
\end{array}
$$
The normal join operator: $R \bowtie S$ will give

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
</tbody>
</table>

whereas $R \bowtie\bowtie S$ will give

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>⊥</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>⊥</td>
</tr>
<tr>
<td>⊥</td>
<td>6</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>

Variation of the outerjoin: $R \bowtie\bowtie_L S$ and $R \bowtie\bowtie_R S$ (left or right outerjoin), that only adds to the normal join the dangling tuples of the left (or right) relation respectively. For the above relations, $R \bowtie\bowtie_L S$ will give

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td>6</td>
<td>⊥</td>
</tr>
<tr>
<td>7</td>
<td>8</td>
<td>9</td>
<td>⊥</td>
</tr>
</tbody>
</table>

and $R \bowtie\bowtie_R S$ will give

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>11</td>
</tr>
<tr>
<td>⊥</td>
<td>6</td>
<td>7</td>
<td>12</td>
</tr>
</tbody>
</table>