Relational Data Model
Overview

- Relational data model;
- Tuples and relations;
- Schemas and instances;
- Named vs. unnamed perspective;
- Relational algebra;
<table>
<thead>
<tr>
<th>Player</th>
<th>Birth Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Andy</td>
<td>1980</td>
</tr>
<tr>
<td>Wim</td>
<td>1975</td>
</tr>
<tr>
<td>Liam</td>
<td>1985</td>
</tr>
<tr>
<td>Mike</td>
<td>1988</td>
</tr>
<tr>
<td>Bert</td>
<td>1971</td>
</tr>
</tbody>
</table>
Observations

- The **rows** of the table contain pairs occurring in the relation **player**.
- There are two **columns**, labeled respectively by “name” and “birth year”.
- The **values** in each column belong to different **domains** of possible values.
How to specify a relation

1. specifying the names of the columns (also called fields or attributes);

2. specifying a domain of possible values for each column;

3. enumerate all tuples in the relation.

(1)–(2) refer to the schema of this relation, while (3) to an instance.
Domains and attributes

- A set of **domains** \( D_1; D_2 \) (sets of values);
- A set of corresponding **domain names** \( d_1, d_2, \ldots \);
- A set of **attributes** \( a_1, a_2, \ldots \).
**Tuple:** any finite sequence \((v_1, \ldots, v_n)\). 

\(n\) is the **arity** of this tuple.

**Relation schema:**

\[ r(a_1:d_1, \ldots, a_n:d_n) \]

where \(n \geq 0\), \(r\) is a relation name, \(a_1, \ldots, a_n\) are distinct attributes, \(d_1, \ldots, d_n\) are domain names.

**Relation instance:**

finite set of tuples \((v_1, \ldots, v_n)\) of arity \(n\) such that \(v_i \in D_i\) for all \(i\).
**Observation**

1. The attributes in each column must be unique.

2. A relation is a set. Therefore, when we represent a relation by a table, the order of rows in the table does not matter.

Let us add to this:

3. The order of attributes does not matter.
New notation for tuples

- A tuple is a set of pairs \{ (a_1, v_1), \ldots, (a_n, v_n) \} denoted by \{ a_1=v_1, \ldots, a_n=v_n \}.

- Let \( d_1, \ldots, d_n \) be domain names and \( D_1; \ldots; D_n \) be the corresponding domains.

- The tuple **conforms** to a relation schema \( r(a_1:d_1, \ldots, a_n:d_n) \) if \( v_i \in D_i \) for all \( i \).
Relational data are structured

Note that in the relational data model tuples stored in a table are **structured**:
- all tuples conform to the same relation schema;
- the values in the same column belong to the same domain.

**Untyped perspective:** there is a single domain, so the second condition can be dropped.
Typed or untyped?

Consider the relation admire:

<table>
<thead>
<tr>
<th>admirer</th>
<th>admired</th>
</tr>
</thead>
<tbody>
<tr>
<td>wim</td>
<td>andy</td>
</tr>
<tr>
<td>mike</td>
<td>wim</td>
</tr>
<tr>
<td>liam</td>
<td>andy</td>
</tr>
<tr>
<td>liam</td>
<td>arsenal</td>
</tr>
</tbody>
</table>

wim andy
mike wim
liam andy
liam arsenal
Database schema and instance


- Relational database instance conforming to a relational database schema \( S \):
  - a mapping \( I \) from the relation names of \( S \) to relation instances such that
    - for every relation schema \( r(a_1:d_1, \ldots, a_n:d_n) \) in \( S \) the relation instance \( I(r) \) conforms to this relation schema.
No attributes

- a tuple is simply a **sequence** \((v_1, \ldots, v_n)\) of values.

- The components of tuples can therefore be identified by their **position** in the tuple.
From Unnamed to Named Perspective

- Introduce a collection of attributes $\#1, \#2, \ldots,$

- Identify tuple $(v_1, \ldots, v_n)$ with the tuple
  \[
  \{ \#1 = v_1, \ldots, \#n = v_n \}.
  \]

- Likewise, identify relation schema $r(d_1, \ldots, d_n)$ with
  \[
  r(\#1:d_1, \ldots, \#n:d_n).
  \]
1. Can **define** new relations from existing ones;
2. Uses a collection of **operations** on relations to do so.
\{ (v_{11}, \ldots, v_{1n}), \\
\ldots. \\
(v_{k1}, \ldots, v_{kn}) \}
\[ R_1 \cup R_2 = \{ (c_1, \ldots, c_k) \mid (c_1, \ldots, c_k) \in R_1 \text{ or } (c_1, \ldots, c_k) \in R_2 \} \]
Set difference

\[ R_1 - R_2 = \{ (c_1, \ldots, c_k) \mid (c_1, \ldots, c_k) \in R_1 \text{ and } (c_1, \ldots, c_k) \notin R_2 \} \]
**Cartesian product**

\[ R_1 \sqsubseteq R_2 = \{(c_1, \ldots, c_k, d_1, \ldots, d_m) \mid (c_1, \ldots, c_k) \in R_1 \text{ and } (d_1, \ldots, d_m) \in R_2 \} \]
Let now $R$ be a relation of arity $k$ and $i_1, \ldots, i_m$ be numbers in \{1, \ldots, k\}.

$$\pi_{i_1, \ldots, i_m}(R) = \{(c_{i_1}, \ldots, c_{i_m}) \mid (c_1, \ldots, c_k) \in R\}.$$ 

We say that $\pi_{i_1, \ldots, i_m}(R)$ is obtained from $R$ by projection (on arguments $i_1, \ldots, i_m$).
Selection

Assume formulas on domains with “variables” \#1, \#2, ….

For example, \#1 = \#2.

\[ \widehat{\pi} (R) = \{(c_1, \ldots, c_k) \mid (c_1, \ldots, c_k) \in R \text{ and } F \text{ holds on } (c_1, \ldots, c_k)\}. \]
Overview

- Relational algebra, named perspective
- SQL
- Integrity constraints
- (Aggregates and grouping)
\{ \{ a_1 = v_{11}, \ldots, a_n = v_{1n} \}, \\
\ldots\quad \ldots\quad \ldots \\
\{ a_1 = v_{k1}, \ldots, a_n = v_{kn} \} \}
Let $R_1$, $R_2$ be relations with the same attributes.

$$R_1 \cup R_2 = \{ t \mid t \in R_1 \text{ or } t \in R_2 \}$$
### Union, example

#### Relations $R_1$ and $R_2$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>†</td>
<td>3</td>
</tr>
</tbody>
</table>

#### Union $R_1 \cup R_2$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>-</td>
<td>2</td>
</tr>
<tr>
<td>3</td>
<td>†</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>†</td>
<td>3</td>
</tr>
</tbody>
</table>
Let $R$ be a relation whose set of attributes is $a_1, \ldots, a_n, c_1, \ldots, c_m$

Let $b_1, \ldots, b_n$ be distinct attributes such that
\[\{b_1, \ldots, b_n\} \sim \{c_1, \ldots, c_m\} = >\]

Then
\[
\sim_{d_4} e_4, \ldots, d_q \sim_e_q (R) =
\{\{b_1 = v_1, \ldots, b_n = v_n, c_1 = w_1, \ldots, c_m = w_m\} | \{a_1 = v_1, \ldots, a_n = v_n, c_1 = w_1, \ldots, c_m = w_m\} \in R\}\]
SQL is based on set and relational operations with certain modifications and enhancements.

A typical SQL query has the form:

```
select a_1, ..., a_n
from R_1, ..., R_m
where P
```

This query is equivalent to relational algebra expression:

```
\pi_{a_1, \ldots, a_n} (\sigma_P (R_1 \times \ldots \times R_m))
```

The result of an SQL query is a relation.

Exceptions?
**Integrity constraints**

- Domain constraints.
- Key constraints.
- Foreign key constraints.

- More general, defined constraints.

- How to translate them?
Query language

Allow one to define:

- Relation and database schemas;
- Relations through our relations;
- Integrity constraints;
- Updates.