SEMINAR SELECTED TOPICS IN DATABASE THEORY

Lecture 1: Introduction / The Relational Model

David Carral, Markus Krötzsch
Knowledge-Based Systems

TU Dresden, 10th October 18
Introduction
Course Tutors

David Carral

Markus Krötzsch
Structure of the Seminar and Evaluation

Lectures

- **Wednesday 10th (i.e., today), DS6:** Introductory lecture 1
- **Wednesday 17th, DS6:** Introductory lecture 2
- **Afterwards:** Office hours in 3035 and presentations

Evaluation

- **Paper summary:** self-selected research paper; a ~15 pages
- **Presentation:**
  - 20 minutes + discussion
  - Participate in the presentations of other students

---

Web Page

Lecture Notes
All slides will be available online.

Reading list
Serge Abiteboul, Richard Hull, Victor Vianu; Foundations of databases. Available at http://webdam.inria.fr/Alice/

Acknowledgements
Check out Vim Martens!
On to the content...
What is a database?

A **Database Management System** (DBMS) is a software to manage collections of data. The **architecture of DBMS** consist of three levels:

- **External Level**: Application-specific user views
- **Logical Level**: Abstract data model, independent of implementation, conceptual view
- **Physical Level**: Data structures and algorithms, platform-specific

**In this seminar**: focus on logical view for relational data model
The Relational Model
Database = collection of tables

Schedule

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodfellas</td>
<td>Thalia</td>
<td>15/10</td>
<td>True</td>
</tr>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
</tr>
<tr>
<td>Annie Hall</td>
<td>Rundkino</td>
<td>21/11</td>
<td>False</td>
</tr>
</tbody>
</table>

A table has a schema:

- Schedule[{{Movie, Cinema, Date, R-rated}}]
Towards a formal definition of “table”

A table row has one value for each column.

- That is, a row is a function from the attributes of the table schema to specific values.

Schedule

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

The above row can be represented with the function:

\[ f : \{ \text{Movie} \mapsto \text{Boogie Nights}, \text{Cinema} \mapsto \text{Rundkino}, \text{Date} \mapsto 21/11, \text{R-rated} \mapsto \text{True}\} \]
Database = set of tables

Let \textbf{dom} (“domain”) be the set of conceivable values in tables.

**Definition 1**

- A relation schema \( R[U] \) consists of a relation name \( R \) and a finite set \( U \) of attributes
- \(|U|\) is the arity of \( R[U] \)
- A table for \( R[U] \) is a finite set of functions from \( U \) to \textbf{dom}
- A database instance \( \mathcal{I} \) is a finite set of tables
Let \textbf{dom} (“domain”) be the set of conceivable values in tables.

\textbf{Definition 1}

- A \textbf{relation schema} $R[U]$ consists of a relation name $R$ and a finite set $U$ of attributes
- $|U|$ is the arity of $R[U]$
- A \textbf{table} for $R[U]$ is a finite set of functions from $U$ to \textbf{dom}
- A \textbf{database instance} $\mathcal{I}$ is a finite set of tables

\textbf{Note:} we disregard the order and multiplicity of rows. Tables are also called relation instances. The table with relation schema $R[U]$ in the database instance $\mathcal{I}$ is written $R^\mathcal{I}$. 
Database = set of tables

Schedule

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodfellas</td>
<td>Thalia</td>
<td>15/10</td>
<td>True</td>
</tr>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
</tr>
<tr>
<td>Annie Hall</td>
<td>Rundkino</td>
<td>21/11</td>
<td>False</td>
</tr>
</tbody>
</table>

- The domain dom of the above table is the following set:
  \{Goodfellas, Thalia, 15/10, True, Unforgiven, Thalia, 17/10, Boogie Nights, Rundkino, 21/11, Annie Hall, Rundkino, False\}

- The above is a table for the relation schema Schedules[\{Movie, Cinema, Date, R-rated\}]

- Let $I$ be a database instance. Then, $Schedules^I$ is the set of rows in this table.
Database = set of tables

**Schedule**

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodfellas</td>
<td>Thalia</td>
<td>15/10</td>
<td>True</td>
</tr>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
</tr>
<tr>
<td>Annie Hall</td>
<td>Rundkino</td>
<td>21/11</td>
<td>False</td>
</tr>
</tbody>
</table>

The table represented above is the set \{r_1, r_2, r_3, r_4\} where \(r_1, r_2,\)
\(r_3,\) and \(r_4\) are the following functions:

\[ r_1 = \{M \mapsto \text{Goodfellas}, C \mapsto \text{Thalia}, D \mapsto 15/10, R \mapsto \text{True}\} \]
\[ r_2 = \{M \mapsto \text{Unforgiven}, C \mapsto \text{Thalia}, D \mapsto 17/10, R \mapsto \text{True}\} \]
\[ r_3 = \{M \mapsto \text{Boogie Nights}, C \mapsto \text{Rundkino}, D \mapsto 21/11, R \mapsto \text{True}\} \]
\[ r_4 = \{M \mapsto \text{Annie Hall}, C \mapsto \text{Rundkino}, D \mapsto 21/11, R \mapsto \text{False}\} \]
Database = set of relations

**Remark:** Attribute names do not matter. Instead of the function

$$\{M \mapsto \text{Goodfellas}, C \mapsto \text{Thalia}, D \mapsto 15/10, R \mapsto \text{True}\}$$

we could also use a tuple:

$$\langle \text{Goodfellas, Thalia, 15/10, True} \rangle$$

**Necessary assumption:** Attributes have a fixed order.

**Definition 2**

- A relation schema $R[U]$ is defined as before
- A table for $R[U]$ is a finite subset of $\text{dom}^{[U]}$
- A database instance $\mathcal{I}$ is a finite set of tables
Database = set of relations

Schedule

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodfellas</td>
<td>Thalia</td>
<td>15/10</td>
<td>True</td>
</tr>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
</tr>
<tr>
<td>Annie Hall</td>
<td>Rundkino</td>
<td>21/11</td>
<td>False</td>
</tr>
</tbody>
</table>

The table represented above is the following set:

\[
\{\langle \text{Goodfellas}, 15/10, \text{True}, \text{Thalia} \rangle, \\
\langle \text{Unforgiven}, 17/10, \text{True}, \text{Thalia} \rangle, \\
\langle \text{Boogie Nights}, 21/11, \text{True}, \text{Rundkino} \rangle, \\
\langle \text{Annie Hall}, 21/11, \text{False}, \text{Rundkino} \rangle\}\]
Another convenient way to write databases:

**Definition 3**
A fact is an expression $p(t_1, \ldots, t_n)$ where

- $p$ is an $n$-ary predicate symbol
- $t_1, \ldots, t_n$ are constant symbols

A database instance is a finite set of facts.
Database = set of facts

**Schedule**

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Goodfellas</td>
<td>Thalia</td>
<td>15/10</td>
<td>True</td>
</tr>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
</tr>
<tr>
<td>Annie Hall</td>
<td>Rundkino</td>
<td>21/11</td>
<td>False</td>
</tr>
</tbody>
</table>

The information in the above corresponds to the following facts:

- Schedule(Goodfellas, 15/10, True, Thalia)
- Schedule(Unforgiven, 17/10, True, Thalia)
- Schedule(Boogie Nights, 21/11, True, Rundkino)
- Schedule(Annie Hall, 21/11, False, Rundkino)
Graphical Representation

Director(Scorsese)
DirectedBy(Goodfellas, Scorsese)
ActsIn(De Niro, Goodfellas)
ActsIn(Pesci, Goodfellas)
<table>
<thead>
<tr>
<th>Perspective</th>
<th>DB Instance</th>
<th>Table</th>
<th>Row</th>
</tr>
</thead>
<tbody>
<tr>
<td>Named</td>
<td>Set of tables</td>
<td>Set of functions</td>
<td>Function</td>
</tr>
<tr>
<td>Unnamed</td>
<td>Set of tables</td>
<td>Set of tuples</td>
<td>Tuple</td>
</tr>
<tr>
<td>Fact-based</td>
<td>Set of facts</td>
<td>Set of facts</td>
<td>Fact</td>
</tr>
<tr>
<td>Graph</td>
<td>Labelled hypergraph</td>
<td>L. hypergraph</td>
<td>L. Edge</td>
</tr>
</tbody>
</table>
The Relational Algebra
Relational Algebra Queries

Query language based on a set of operations on databases. Each operation refers to some tables and produces another table.

Main operations of the named perspective:

- Selection $\sigma$
- Projection $\pi$
- Join $\bowtie$
- Renaming $\delta$
- Difference $-$
- Union $\cup$
- Intersection $\cap$
Selection

“Find all R-rated movies”

\( \sigma_{\text{R-rated}=\text{"True"}} \) Schedule

“Find all connections that begin and end in the same stop”

\( \sigma_{\text{From}=\text{to}} \) Connect

Definition 4

The **selection operator** has the form \( \sigma_{n=m} \)

- \( n \) is an attribute name
- \( m \) is an attribute name or a constant value

Consider a table \( R^I \) for the relational schema \( R[U] \).

- For \( m \) constant value: \( \sigma_{n=m}(R^I) = \{ f \in R^I \mid f(n) = m \} \)
- For \( m \) constant value: \( \sigma_{n=m}(R^I) = \{ f \in R^I \mid f(n) = f(m) \} \)
Selection

“Find all dates in which some movie is projected.”

\[ \pi_{\text{Date Schedule}} \]

**Definition 5**

The **projection operator** has the form \( \pi_{a_1,\ldots,a_n} \) where each \( a_i \) is an attribute name.

Consider a table \( R^I \) for \( R[U] \).

\[ \pi_{a_1,\ldots,a_n}(R^I) = \{ f_{\{a_1,\ldots,a_n\}} \mid f \in R^I \} \]

where \( f_{\{a_1,\ldots,a_n\}} \) is the restriction of \( f \) to the domain \( \{a_1,\ldots,a_n\} \), i.e., the function \( \{a_1 \mapsto f(a_1), \ldots, a_n \mapsto f(a_n)\} \).

**Remark:** Projection is only defined if \( a_i \in U \) for each \( a_i \).
## Natural Join

### Schedule

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
<th>Neighborhood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
<td>Neudstadt</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
<td>Altstadt</td>
</tr>
</tbody>
</table>

### Location

<table>
<thead>
<tr>
<th>Cinema</th>
<th>Neighborhood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thalia</td>
<td>Neudstadt</td>
</tr>
<tr>
<td>Rundkino</td>
<td>Altstadt</td>
</tr>
</tbody>
</table>

### Schedule $\bowtie$ Location

<table>
<thead>
<tr>
<th>Movie</th>
<th>Cinema</th>
<th>Date</th>
<th>R-rated</th>
<th>Neighborhood</th>
</tr>
</thead>
<tbody>
<tr>
<td>Unforgiven</td>
<td>Thalia</td>
<td>17/10</td>
<td>True</td>
<td>Neudstadt</td>
</tr>
<tr>
<td>Boogie Nights</td>
<td>Rundkino</td>
<td>21/11</td>
<td>True</td>
<td>Altstadt</td>
</tr>
</tbody>
</table>
Natural Join

**Definition 6**
The **natural join** operator has the form $\sqcap\Join$.
Consider tables $R^I$ for $R[U]$ and $S^I$ for $S[V]$.

$$R^I \Join S^I = \{ f : U \cup V \rightarrow \text{dom} | f_U \in R^I \text{ and } f_V \in S^I \}$$

where $f_U$ (resp. $f_V$) is the restriction of $f$ to elements in $U$ (resp. $V$) as before.
Rename

\[ \delta_{\text{Movie}, \text{Cinema}, \text{Date}, \text{R-rated}} \rightarrow \text{Film}, \text{Cinema}, \text{Date}, \text{R-rated} (\text{Schedule}) \]

**Definition 7**

The **renaming operator** has the form \( \delta_{a_1, \ldots, a_n \rightarrow b_1, \ldots, b_n} \) with all \( a_i \) mutually distinct attribute names, and likewise for all \( b_i \).

Consider a table \( R^I \) for \( R[[a_1, \ldots, a_n]] \)

\[
\delta_{a_1, \ldots, a_n \rightarrow b_1, \ldots, b_n} (R^I) = \{ f \circ g \mid f \in R^I \text{ and } g : \{ b_i \mapsto a_i \}_{1 \leq i \leq n} \}
\]

where \( f \circ g \) is function composition: \( (f \circ g)(x) = f(g(x)) \)
Difference, Union, Intersection

- Binary operators defined like the usual set operations.
- **Remark:** These operators are only defined on tables of the same relational schema. That is, tables with the same set of attributes.
Research on Database Theory

• Relational algebra query answering.
  – Studying combined/data/query Complexity.
  – Fragments of relational algebra such as conjunctive queries. That is, relational algebra expressions that use only select, project, join, and rename.
  – Extensions of relational algebra such as Datalog.
Research on Database Theory

- Relational algebra query answering.
  - Studying combined/data/query Complexity.
- Developing and studying novel query languages.
  - Fragments of relational algebra such as conjunctive queries. That is, relational algebra expressions that use only select, project, join, and rename.
  - Extensions of relational algebra such as Datalog.
Research on Database Theory

• Relational algebra query answering.
  – Studying combined/data/query Complexity.
• Developing and studying novel query languages.
  – Fragments of relational algebra such as conjunctive queries. That is, relational algebra expressions that use only select, project, join, and rename.
  – Extensions of relational algebra such as Datalog.
• Study the “expressivity” of a query language.
Research on Database Theory

- Relational algebra query answering.
  - Studying combined/data/query Complexity.

- Developing and studying novel query languages.
  - Fragments of relational algebra such as conjunctive queries. That is, relational algebra expressions that use only select, project, join, and rename.
  - Extensions of relational algebra such as Datalog.

- Study the “expressivity” of a query language.
  - Let’s do this!