THE RELATIONAL MODEL

University of Waterloo
List of Slides

1
2 The Relational Model
3 Relations and Databases
4 Example
5 Another Example
6 What does it mean?
7 Example Database
8 What can we do with it?
9 Variables and Answers
10 . . . its just like Logic
11 Variables and Quantifiers
12 Selecting Things
13 Combining Relations
14 Negation and CWA
15 Safety
16 Relational Calculus
17 Properties of RC
18 Summary
The Relational Model

Entities:

(names of) real-world objects
⇒ “customer”, “bank”, “account”, ...
⇒ “author”, “publication”, ...
⇒ “1” (the number 1), ...

Relationships:

relationships between entities
⇒ “has an account”, “located at”, ...
⇒ “wrote”, “author”, “book”, ...

Attributes:

properties of relationships (and, in turn, entities)
⇒ “address”, “account #”, “balance”, ...
⇒ “name”, “title”, ...
A collection of *data* describing a real-world situation.

Entities are (generally) “real world” things we want to store information about. In the database they’re encoded using an identifier.

Relationships tie entities with their other entities.

Attributes form the “observable” properties of relationships; note that two tuples with identical values of attributes cannot be distinguished and are considered to be the same; that’s why everyone has a SSN (SIN in Canada, CPR# in Denmark).

In the above example, “author” is an entity. However, to associate author names and id’s with the entities, we can use an “author” relationship.
Relations and Databases

Idea: All information is organized in relations.

A RELATION has

- a name
- a schema
  - a list of attribute names describing individual elements of tuples (and their types)
- an instance
  - a finite set of tuples of (identifiers of) entities

The attribute name denotes what role an entity plays in a relation(ship).
## Example

### publications

<table>
<thead>
<tr>
<th>pubid</th>
<th>title</th>
<th>author</th>
<th>type</th>
<th>year</th>
<th>cref</th>
</tr>
</thead>
<tbody>
<tr>
<td>ChSa98</td>
<td>Logics for Databases and Information Systems</td>
<td>Chomicki, Jan</td>
<td>book</td>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>ChTo98</td>
<td>Temporal Logic in Information Systems</td>
<td>Chomicki, Jan</td>
<td>chapter</td>
<td>1998</td>
<td>ChSa98</td>
</tr>
<tr>
<td>ChTo98</td>
<td>Temporal Logic in Information Systems</td>
<td>Toman, David</td>
<td>chapter</td>
<td>1998</td>
<td>ChSa98</td>
</tr>
<tr>
<td>JLP0398</td>
<td>Journal of Logic Programming</td>
<td></td>
<td>journal</td>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>ChTo98a</td>
<td>Datalog with Integer Periodicity Constraints</td>
<td>Toman, David</td>
<td>article</td>
<td></td>
<td>JLP0398</td>
</tr>
<tr>
<td>DOOD97</td>
<td>International Conference on Deductive and Object-Oriented Databases</td>
<td></td>
<td>proceedings</td>
<td>1998</td>
<td></td>
</tr>
<tr>
<td>Tom97</td>
<td>Point-Based Temporal Extension of Temporal SQL</td>
<td>Toman, David</td>
<td>paper</td>
<td></td>
<td>DOOD97</td>
</tr>
</tbody>
</table>
Another Example

<table>
<thead>
<tr>
<th>author</th>
<th>wrote</th>
</tr>
</thead>
<tbody>
<tr>
<td>aid</td>
<td>Name</td>
</tr>
<tr>
<td>1</td>
<td>Toman, David</td>
</tr>
<tr>
<td>2</td>
<td>Chomicki, Jan</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>publication</th>
<th>publication</th>
</tr>
</thead>
<tbody>
<tr>
<td>pubid</td>
<td>title</td>
</tr>
<tr>
<td>ChTo98</td>
<td>Temporal Logic in Information Systems</td>
</tr>
<tr>
<td>JLP-3-98</td>
<td>Journal of Logic Programming</td>
</tr>
<tr>
<td>ChTo98a</td>
<td>Datalog with Integer Periodicity Constraints</td>
</tr>
<tr>
<td>DOOD97</td>
<td>International Conference on Deductive and Object...</td>
</tr>
<tr>
<td>Tom97</td>
<td>Point-Based Temporal Extension of Temporal SQL</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
What does it mean?

Membership (presence) of a tuple \((a_1, \ldots, a_k)\) in a relation \(R\) means that the relationship \(R\) holds among \((a_1, \ldots, a_k)\) in the given database instance \(D\).

We write \((a_1, \ldots, a_k) \in R^D\) or \(D \models R(a_1, \ldots, a_k)\).

For example:

\[
\text{DB} \models \text{wrote}(1, \text{ChTo98})
\]

means that, in the database \(\text{DB}\), a publication identified by \(\text{ChTo98}\) has been written by author with author id 1.

Database modification (update) changes the relationships that are true/false in the database (for which a tuple is present/absent).
Example Database

Tables (in the DB database) used in examples:

author(aid, name)
wrote(author, publication)
publication(pubid, title)
book(pubid, publisher, year)
journal(pubid, volume, no, year)
proceedings(pubid, year)
article(pubid, crossref, startpage, endpage)
What can we do with it?

Idea: Combine simple questions about what is true in the database into more complex questions using logical connectives.

- Yes/No queries:
  - in $\text{DB}$, does the publication $\text{ChTo98}$ have a title "Temporal Logic in Information Systems"?
  - ... we could explore the database by asking (lots) of Yes/No queries.
    - not efficient
    - you can only verify statements
  - ... not quite satisfactory:
    - we'd like to retrieve all elements for which some statement holds “in one shot”.
Variables and Answers

Idea: we use logical variable(s) to range over values in the database. The values of these variable(s) form new relations.

- variables and answers:
  - Yes/No queries are parametrized by variables.
  - An answer is the set of all values that, when substituted for the variables, make the query true in \( D \).
  - retrieve all \( x \) such that \( Q(x) \) is true in \( D \):
    \[
    \{ x : D \models Q(x) \}
    \]
  - queries may contain many parameters (variables).
...its just like Logic

Don’t panic!

• Statements about numbers:
  \[ \Rightarrow 1 < 3 \text{ is true (in } \mathbb{R}) \]
  \[ \Rightarrow 2 + 2 = 0 \text{ is false in } \mathbb{Z} \text{ (but true in } \mathbb{Z}_4) \]
  \[ \Rightarrow \exists x. x^2 = -1 \text{ (false in } \mathbb{R}, \text{ true in } \mathbb{C}) \]

• Formation of sets:
  \[ \Rightarrow \{ x : x > 5 \} \text{ the set of all numbers bigger than 5} \]
  \[ \Rightarrow \{ x : x^2 - 3x + 2 = 0 \} \text{ the roots of the equation} \]
  \[ \Rightarrow \{ (x, y) : x = \sin(t) \land y = \cos(t), t \in \mathbb{R} \} \text{ a circle} \]
Variables and Quantifiers

Often we don’t care what value(s) a variable contains, as long as there is at least one such value.

Moreover, we often do not want such values to appear in the answer.

- Questions of the form “there is a value for $x$ such that $Q(x)$ is true in the database”:

  $$\{y : \text{DB} \models \exists x. Q(x, y)\}$$

  note that $Q$ may contain other variables (e.g., $y$).
Selecting Things

If we need to retrieve only answers that satisfy some additional predicate we use **built-in** predicates.

- equality: \( x = y, x = 5, x = '\text{string}' \), etc.

- other predicates (depending on the data type, e.g., \( x > 5 \) for numerical values)

The built-in predicates also define tables, but are not *stored* in the database:

⇒ the DBMS computes the value of the predicate.
⇒ the arguments have to be **bound** in a base relation.
Combining Relations

We can ask about what is true in several relations:

- conjunction ($\land$) of formulas (usually atomic)

- use of shared variables:
  $\Rightarrow R(x, y) \land S(y, z)$
  $\Rightarrow$ forces equality on parts of tuples.

- use of quantification to hide uninteresting variables:
  $\Rightarrow \exists y. R(x, y) \land S(y, z)$

- by far the most common pattern in queries
Negation and CWA

**Idea:** Only relationships explicitly stored in the database are true.

- everything not in the database is false:
  - to find out that

  \[ \text{DB} \models \neg \text{wrote}(5, \text{ChTo98}) \]

  we simply see that

  \[ \text{DB} \not\models \text{wrote}(5, \text{ChTo98}) \]

  i.e., the tuple (5, ChTo98) is not in \text{wrote}.

- Closed World Assumption.
Safety

- **Answers** to a queries have to be **finite**.
- Not completely clear for negation, compare the following two queries:
  - ⇒ List the names of all authors (good).
  - ⇒ List the names of all non-authors (bad).
- Only **Range-Restricted** queries allowed
  - all SQL queries are range-restricted.
Relational Calculus

The language we formulate statements about truth in the database is based on **first-order (predicate) logic** (with few restrictions that guarantee safety):

\[
Q ::= R(x_1, \ldots, x_k) \\
| Q_1 \land Q_2 \\
| \exists x_i . Q \\
| Q \land (x_i = x_j) \quad \text{for } \{x_i, x_j\} \subseteq FV(Q) \\
| Q_1 \land \neg Q_2 \quad \text{for } FV(Q_2) \subseteq FV(Q_1) \\
| Q_1 \lor Q_2 \quad \text{for } FV(Q_1) = FV(Q_2)
\]

where \( FV(Q) \) is the set of **free variables** of \( Q \).
Properties of RC

- Clean semantics
  ⇒ Results do **NOT** depend on implementation
  ⇒ Efficient implementation (Relational Algebra)

- All queries terminate
  ⇒ not Turing complete (in PTIME, LOGSPACE)
  ⇒ estimation of execution time possible

- Expressiveness results
  ⇒ it is known what questions can be formulated in RC
    sufficient for most business applications,
    but no transitive closure, cardinality, counting, . . .
Summary

- A **Relational database** stores information about **relationships** between entities in **finite relations**.
  - Entities are denoted by constants *which the DBMS does NOT understand*.
  - A relationship can be **true**: a tuple is present, or **false**: a tuple is absent (CWA).

- A **query language** allows us to combine simple facts into complex statements:
  - The statements may be parametric:
    - The result is a set of values that, when substituted for parameters, make the statement true.
  - Results of queries are relations themselves.

- We use (safe) **relational calculus** to formulate queries. SQL is just a (strange) syntax for RC.