The Relational Model
An Informal Introduction

Olaf Hartig
David R. Cheriton School of Computer Science
University of Waterloo

CS 640
Principles of Database Management and Use
Winter 2013

These slides are based on a slide set provided by Prof. M. Tamer Özsu.

The Relational Model

- was proposed in 1969 by Edgar F. Codd;
- is a data model for the logical level, that
  - comes with a well-defined theory, and
  - abstracts from any particular implementation details;
- supports a high degree of data independence;
- supports semantic integrity constraints;
- supports powerful and declarative query/update languages;
- organizes data in flat, table-like structures (called relations).

A Relation (Informally)

<table>
<thead>
<tr>
<th>Musician</th>
<th>Instrument</th>
<th>Band</th>
<th>From</th>
<th>Until</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ringo Starr</td>
<td>Drums</td>
<td>The Beatles</td>
<td>1962</td>
<td>1970</td>
</tr>
<tr>
<td>Paul McCartney</td>
<td>Bass</td>
<td>The Beatles</td>
<td>1961</td>
<td>1970</td>
</tr>
<tr>
<td>Paul McCartney</td>
<td>Guitar</td>
<td>The Quarrymen</td>
<td>1957</td>
<td>1960</td>
</tr>
<tr>
<td>Lars Frederiksen</td>
<td>Guitar</td>
<td>Rancid</td>
<td>1993</td>
<td>present</td>
</tr>
</tbody>
</table>

Relation Scheme: \( R(A_1, A_2, \ldots, A_k) \) with
- \( R \) is the relation name
- \( A_1, \ldots, A_k \) are attributes (with distinct names)

Relation Instance: a set \( I \subseteq \text{dom}(A_1) \times \cdots \times \text{dom}(A_k) \)
- Each element \( u \in I \) is called a tuple.

Database Scheme: set of uniquely-named relation schemes
Database Instance: a relation instance for each relation scheme
Properties of Relation Schemes and Instances

Degree: Number of attributes in scheme (also called arity)
Cardinality: Number of tuples in instance

- Attribute ordering: not strictly necessary
- Value oriented: tuples identified by attribute values
- Instance has set semantics:
  - No ordering among tuples
  - No duplicate tuples

First Normal Form (1NF): All attribute values are atomic

Integrity Constraints

A relational scheme captures only the structure of relations

Idea
When we specify a scheme, we specify additional conditions—called integrity constraints—that must hold for any possible instance.

- Primary key constraints
- Foreign key constraints
- Functional dependencies
- etc.

Primary Key Constraints (Informally)

Superkey: a set of attributes for which no pair of distinct tuples in the relation will ever agree on the corresponding values

(Candidate) Key: a minimal superkey (a minimal set of attributes that uniquely identifies a tuple)

Primary Key: a designated candidate key

Example:

author( aid, name )
wrote( author, publication )
publication( pubid, title )
**Foreign Key Constraints (Informally)**

**Foreign key:** Attribute(s) of one relation used for referring to tuples in another relation (i.e. a "logical pointer").
- Foreign key attribute(s) must correspond to primary key attribute(s) of the referenced relation
- Attribute names may differ

**Referential integrity:** Tuples with a foreign key that does not match the primary key of a tuple in the referenced relation are not allowed.

**Example:**

```sql
author( aid , name )
wrote( author, publication )
FOREIGN KEY (author) REFERENCES author(aid)
FOREIGN KEY (publication) REFERENCES publication(pubid)
publication( pubid, title )
```

---

**Functional Dependencies (Informally)**

**Functional Dependency (FD):** $X \rightarrow Y$ requires that if two tuples agree on the values for attributes in $X$, they must also agree on the values for attributes in $Y$.

**Example:**
- Consider a scheme: EmpProj( SIN, PNum, EName, FName, Loc )
- $\text{SIN}$ functionally determines employee name:
  - $\{ \text{SIN} \} \rightarrow \{ \text{EName} \}$
  - (i.e. tuples with the same SIN must also have the same EName)
- Project number determines project name and location:
  - $\{ \text{PNum} \} \rightarrow \{ \text{ FName, Loc} \}$

**Note**

Functional dependencies can be used to identify (candidate) keys.

---

**Functional Dependencies (cont’d)**

**Relation Schema:** A relation scheme $R(A_1, A_2, \ldots, A_n)$ together with a set of functional dependencies defined over this scheme.

**Database Schema:** set of uniquely-named relation schemas
Integrity Constraints Revisited

- A database instance is invalid if it violates an integrity constraint
- DBMS should enforce integrity constraints

Note
The specification of integrity constraints is a design decision.

Advantages of using integrity constraints:
- Ensure data entry/modification respects database design
  - Shift responsibility from applications to DBMS
- Protect a database from bugs in applications

Relations vs. SQL Tables

Note
The standard language for interfacing with relational DBMSs is Structured Query Language (SQL). Unfortunately, there are a few important differences between the Relational Model and the data model used by SQL (and relational DBMSs).

Discrepancies between the relational model and the SQL data model:
1. Semantics of Instances
   - Relations are sets of tuples
   - SQL tables are multisets (bags) of tuples (i.e. duplicates possible)
2. Unknown values
   - SQL data model defines a particular value called null (intended to mean ‘unknown’) which has some special properties (requires three-value logic)
   - no such thing in (the pure) relational model

Summary and Outlook

- Basic structural elements:
  - relation scheme, attributes, attribute domains
  - relation instance, tuples, attribute values
- Integrity constraints
  (conditions that must be satisfied by any valid instance)
  - primary key constraints (superkey, candidate key, primary key)
  - foreign key constraints
  - functional dependencies
  - others; e.g. multivalued dependencies (not covered here)
- Next week:
  - a formal view on the relational model
  - basics of (functional) dependency theory