

# The Relational Model

Grant Weddell

Cheriton School of Computer Science  
University of Waterloo

CS 348  
Introduction to Database Management  
Winter 2017

# Review: Network and Hierarchical Models

## Idea

*Structural information is encoded by owner-coupled sets; querying by iterating over such sets.*

## Idea

*Structural information is encoded by owner-coupled sets; querying by iterating over such sets.*

## Consequences:

- technology often conflates owner-coupled sets with concrete data structures
  - ⇒ introduces physical data dependence
- a query can only navigate such sets
  - ⇒ *procedural* queries
  - ⇒ *procedural* specification of *integrity constraints*

# The Relational Model

## Idea

*Use predicate logic: a database schema is based on a relational signature; well formed formulas and sentences are queries and schema integrity constraints.*

# The Relational Model

## Idea

*Use predicate logic: a database schema is based on a relational signature; well formed formulas and sentences are queries and schema integrity constraints.*

## Features:

- simple and clean data model
- powerful and declarative query/update language(s)
- powerful and declarative integrity constraints
- data independence more officially sanctioned

# The Relational Model: Formal Definition

- Universe** • a set of atomic values  $\mathbf{D}$  with equality ( $=$ )
- Domain** • a name  $D$  with a set of values  $\text{dom}(D) \subseteq \mathbf{D}$
- Relation** • **schema:**  $R(A_1 : D_1, A_2 : D_2, \dots, A_k : D_k)$  with
- name  $R$
  - $A_1, \dots, A_k$  a set of distinct attribute names
  - $D_1, \dots, D_k$  a collection of (not necessarily distinct) domain names
- **instance:** a **finite** relation  
 $\mathbf{R} \subseteq \text{dom}(D_1) \times \dots \times \text{dom}(D_k)$ .
- Database** • **schema:** finite set of uniquely-named relation schemas
- **instance:** a relation  $\mathbf{R}_i$  for each  $R_i$

## Note

- **Intention of a relation:** *The associated relation schema.*
- **Extension of a relation:** *The associated set of tuples.*

# The Relational Model: Properties

## Note

- *Relational schemas have named and typed attributes*
- *Relational instances are finite*

# The Relational Model: Properties

## Note

- *Relational schemas have named and typed attributes*
- *Relational instances are finite*

Properties of a relation:

- 1 Based on (finite) set theory
  - Attribute ordering: not strictly necessary
  - Value oriented: references to things in terms of values
  - Instance has *set semantics*:
    - No ordering among tuples
    - No duplicate tuples
- 2 All attribute values are atomic (*first normal form*)
- 3 Degree (arity) = # of attributes in schema
- 4 Cardinality = # of tuples in instance



# Example: A Bibliography Database

Database schema:

```
author(aid:int, name:string)
wrote(author:int, publication:int)
publication(pubid:int, title:string)
book(pubid, publisher, year)
journal(pubid, volume, no, year)
proceedings(pubid, year)
article(pubid, crossref, startpage, endpage)
```

## Note

*Relational schemas are sometimes abbreviated by omitting the attribute domains.*

# Example: A Bibliography Database

Sample database instance:

**author** = { (1, John), (2, Sue) }  
**wrote** = { (1, 1), (1, 4), (2, 3) }  
**publication** = { (1, Mathematical Logic),  
(3, Trans. Databases),  
(2, Principles of DB Syst.),  
(4, Query Languages) }  
**book** = { (1, AMS, 1990) }  
**journal** = { (3, 35, 1, 1990) }  
**proceedings** = { (2, 1995) }  
**article** = { (4, 2, 30, 41) }

# Example: A Bibliography Database

Sample database instance (tabular form):

author

aid	name
1	John
2	Sue

wrote

author	publication
1	1
1	4
2	3

publication

pubid	title
1	Mathematical Logic
3	Trans. Databases
2	Principles of DB Syst.
4	Query Languages

## 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 Relational Model and SQL:

## 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 Relational Model and SQL:

### ① Semantics of Instances

- Relations are **sets** of tuples
- Tables are **multisets (bags)** of tuples

## 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 Relational Model and SQL:

### ① Semantics of Instances

- Relations are **sets** of tuples
- Tables are **multisets (bags)** of tuples

### ② Null Values

- SQL data model defines a particular value “null” (intended to mean “no information”) which has some special properties (requires *three-value logic*)

# Integrity Constraints

A relational schema captures only the structure of relations

# Integrity Constraints

A relational schema captures only the structure of relations

## Idea

*Extend relational/database schema with special varieties of sentences called constraints. An instance is only valid if it satisfies all schema constraints.*



# Integrity Constraints

A relational schema captures only the structure of relations

## Idea

*Extend relational/database schema with special varieties of sentences called constraints. An instance is only valid if it satisfies all schema constraints.*

Reasons to use constraints:

- 1 Ensure data entry/modification respects database design
  - Shift responsibility from applications to DBMS
- 2 Protect data from bugs in applications

# Types of Integrity Constraints

- Tuple-level
  - Domain restrictions
  - Attribute comparisons

# Types of Integrity Constraints

- Tuple-level
  - Domain restrictions
  - Attribute comparisons
- Relation-level
  - Key constraints
    - **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
  - Functional dependencies, etc.

# Types of Integrity Constraints (cont'd)

- Database-level
  - Referential integrity
    - **Foreign key**: Primary key of one relation appearing as attributes of another relation.
    - **Referential integrity**: A tuple with a non-null value for a foreign key that does not match the primary key value of a tuple in the referenced relation is not allowed.
  - Inclusion dependencies

# Example: Database Schema showing ICs

