CS 44800: Introduction To Relational Database Systems

Prof. Chris Clifton

26 August 2021

Relational Model

Data Models: Example

- Consider the example of employees and departments:
  - **Entities**: John Smith, the department #30
  - **Entity sets**: the set of all employees, the set of all departments
  - **Attributes**: employee name, salary, job, department number, department name
  - **Relationships**: the fact that John Smith works in the department #30

Purdue University
Department of Computer Science

The Relational Data Model

- Based on a single data structure – the *relation*
- A relation can be seen as a table
  - rows, called *tuples*
  - columns containing values of specific types, such as integer numbers or strings

Purdue University
Department of Computer Science

Tables
representation of relationships

Courses

Course-Name	Instructor	Room-Name
Databases	Smith	DS1
Operating Syst.	Jones	N3
Networks	Li	N3
Security	Li	G

Rooms

Room-Name	Building	Floor
DS1	Recitation	1
N3	Recitation	1
G	Univ. Hall	2

PURDUE UNIVERSITY
Department of Computer Science

DBMS: languages

- **Data Definition Language** (DDL). It allows one to define:
  - The logical schema of the DB
  - The semantic integrity constraints
  - The authorizations for data accesses
- **Data Manipulation Language** (DML)
  - Used for data retrieval (query language) and for data updates
- **Storage Definition Language** (SDL)
  - Used to define physical access structures

Relation Schema and Instance

- $A_1, A_2, \ldots, A_n$ are *attributes*
- $R = (A_1, A_2, \ldots, A_n)$ is a *relation schema*

  Example:

  instructor = (*ID*, *name*, *dept_name*, *salary*)
- A relation instance *r* defined over schema *R* is denoted by *r* (*R*).
- The current values a relation are specified by a table
- An element ***t*** of relation ***r*** is called a *tuple* and is represented by a *row* in a table

Attributes

- The set of allowed values for each attribute is called the **domain** of the attribute
- Attribute values are (normally) required to be **atomic**; that is, indivisible
- The special value ***null*** is a member of every domain. Indicated that the value is "unknown"
- The null value causes complications in the definition of many operations

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- Example: *instructor* relation with unordered tuples

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000

Database Schema

- Database schema -- is the logical structure of the database.
- Database instance -- is a snapshot of the data in the database at a given instant in time.
- Example:
  - schema: *instructor* (*ID, name, dept_name, salary*)
  - Instance:

ID	*name*	*dept_name*	*salary*
22222	Einstein	Physics	95000
12121	Wu	Finance	90000
32343	El Said	History	60000
45565	Katz	Comp. Sci.	75000
98345	Kim	Elec. Eng.	80000
76766	Crick	Biology	72000
10101	Srinivasan	Comp. Sci.	65000
58583	Califieri	History	62000
83821	Brandt	Comp. Sci.	92000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
76543	Singh	Finance	80000

Keys

- Let K $\subseteq$ R
- *K* is a **superkey** of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)*
  - Example: {*ID*} and {ID,name} are both superkeys of *instructor.*
- Superkey *K* is a **candidate key** if *K* is minimal
  Example: {*ID*} is a candidate key for *Instructor*
- One of the candidate keys is selected to be the **primary key**.
  - Which one?
- **Foreign key** constraint: Value in one relation must appear in another
  - **Referencing** relation
  - **Referenced** relation
  - Example: *dept_name* in *instructor* is a foreign key from *instructor* referencing *department*

PURDUE UNIVERSITY
Department of Computer Science

Relational Database Design

- Multiple ways to represent data
  - Which is correct?
- Options enforce *constraints*
  - Different sections with different instructors vs. same instructor across all sections
  - Room/time required vs. optional

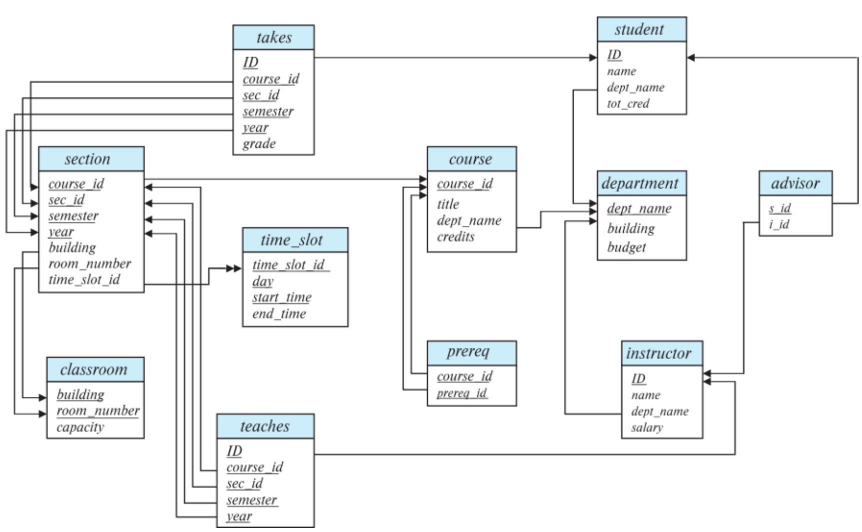
Course	Instructor	Time	Room
CS44800	Clifton	MWF 9:30	PHYS 114
CS34800	Benotman	Asynchronous	

Course	Instructor
CS44800	Clifton
CS34800	Benotman

Course	Time	Room
CS44800	MWF 9:30	PHYS 114

26

Schema Diagram for University Database



Database System Concepts - 7th Edition   2.27   ©Silberschatz, Korth and Sudarshan

Relational Database Design

- There is a solid theory behind a good database design
  - Based on the concept of *keys* and *functional dependencies*
- **Key**: a given attribute value is unique in the relation
  - E.g., In the courses/rooms/times, each room/time only appears once
- **Functional Dependency**: An attribute value in one relation MUST have a corresponding value in another relation

A good relational database design makes it easy to ensure keys and dependencies hold

- More on this later