RELATIONAL DATA MODEL

Fei Chiang (fchiang@mcmaster.ca)

Describing Data: Data Models

- A *data model* is a collection of concepts for describing data.
- A *schema* is a description of a particular collection of data, using a given data model.
- The *relational model of data* is the most widely used model today.
  - Main concept: *relation*, basically a table with rows and columns.
  - Use tables to represent data and relationships
  - Every relation has a *schema*, which describes the columns, or attributes.

Acknowledgement: Renee J. Miller

Relational Model

- Proposed by Edgar. F. Codd in 1970 as a data model which strongly supports data independence.
- Made available in commercial DBMSs in 1981 -- it is not easy to implement data independence efficiently and reliably!
- It is based on (a variant of) the mathematical notion of ***relation***.
- Relations are represented as tables.

A relation is a table

Relation Name

Students

Attribute Names

Tuples (Records)

ID	Name
2225555	Peter Jones
1234567	Amber Smith

The set of permitted values for an attribute is called the attribute ***domain.***
E.g., domain(ID) = {2225555, 1234567}.

Relational Data Model

- *Relation schema* = relation name and attribute list.
  - Optionally: types of attributes. For example:
  - *Students(id, name)*
  - *Students(id: string, name: string)*
- *Relation* = set of tuples conforming to schema
  - Example:
    - { (2225555, Peter Jones), (1234567, Amber Smith), …}
- *Database* = set of relations.
- *Database schema* = set of all relation schemas in the database.

Why Relations?

- Very simple model.
- *Often* matches how we think about data.
- Abstract model that underlies SQL, the most important database language today.

Relations are Unordered

- Order of tuples is irrelevant (tuples may be stored in an arbitrary order)
- E.g., *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
76543	Singh	Finance	80000

Database

- Information about an enterprise is broken up into parts

  instructor
  student
  advisor

- Bad design:
  univ (instructor_ID, name, dept_name, salary, student_Id, ..)
  results in
  - repetition of information (e.g., two students have the same instructor)
  - the need for NULL values (e.g., represent an student with no instructor)

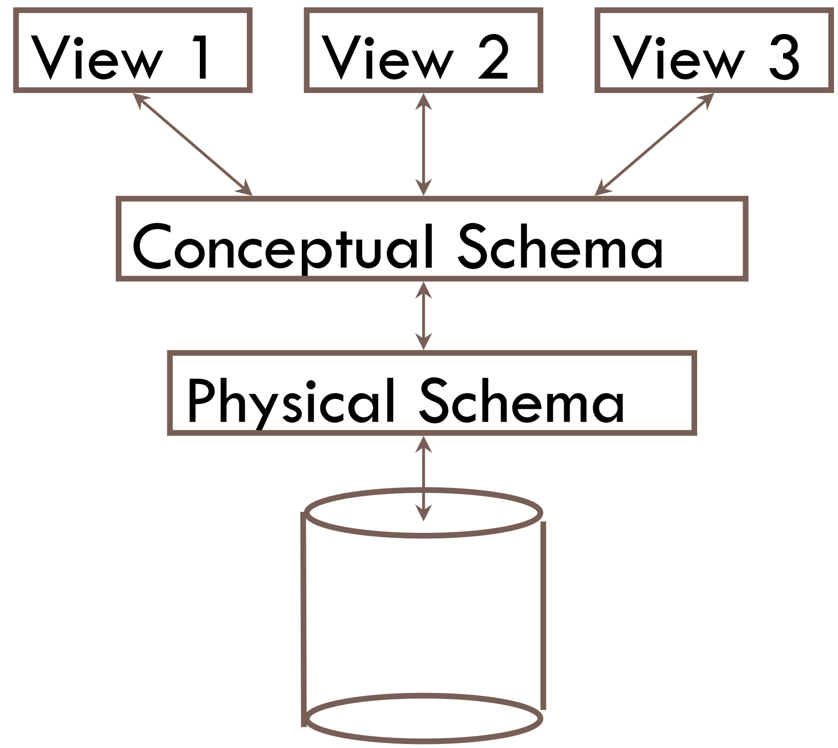
- Normalization theory deals with how to design "good" relational schemas

Database Schemas in SQL

- SQL is primarily a query language, for getting information from a database.
- But SQL also includes a *data-definition* component for describing database schemas.

Levels of Abstraction

- Many *views*, single *conceptual (logical) schema* and *physical schema*.
  - Views describe how users see the data.
  - Conceptual schema defines logical structure
  - Physical schema describes the files and indexes used.



☛ *Schemas are defined using DDL (data definition language);*
☛ *Data is modified/queried using DML (data manipulation language).*

Example: University Database

- Conceptual schema:
  - *Students(sid: string, name: string, login: string, age: integer, gpa:real)*
  - *Courses(cid: string, cname:string, credits:integer)*
  - *Enrolled(sid:string, cid:string, grade:string)*
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):
  - *Course_info(cid:string,enrollment:integer)*

Integrity Constraints

- An *integrity constraint* is a property that must be satisfied by all meaningful database instances.
- A constraint can be seen as *a predicate*; a database is *legal* if it satisfies all integrity constraints.
- Types of constraints
  - Intra-relational constraints: e.g., *domain constraints* and *tuple constraints*
  - Inter-relational constraints: most common is *referential constraint*

Tuple and Domain Constraints

- A *tuple constraint* expresses conditions on the values of each tuple, independently of other tuples.
- E.g., **`Net = Amount-Deductions`**

- A *domain constraint* is a tuple constraint that involves a single attribute
- e.g., **`(GPA ≤ 4.0) AND (GPA ≥ 0.0)`**

Unique Values for Tuples

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

- Registration number identifies students, i.e., there is no pair of tuples with the same value for **`RegNum`**.
- Personal data could identify students as well, i.e., there is no pair of tuples with the same values for all of **`Surname`**, **`FirstName`**, **`BirthDate`**.

Keys

- A *key* is a set of attributes that uniquely identifies tuples in a relation.
- More precisely:
  - A set of attributes K is a *superkey* for a relation r if r cannot contain two distinct tuples t_1 and t_2 such that $t_1[K]=t_2[K]$;
  - K is a (candidate) *key* for r if K is a minimal superkey

(that is, there exists no other superkey K' of r that is contained in K as proper subset, i.e, $K' \subset K$)

Example

RegNum	Surname	FirstName	BirthDate	DegreeProg
284328	Smith	Luigi	29/04/59	Computing
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Fine Art

- **RegNum** is a key: i.e., **RegNum** is a superkey and it contains a sole attribute, so it is minimal.
- **{Surname, Firstname, BirthDate}** is another key

Beware!

RegNum	Surname	FirstName	BirthDate	DegreeProg
296328	Smith	John	29/04/59	Computing
587614	Smith	Lucy	01/05/61	Engineering
934856	Black	Lucy	01/05/61	Fine Art
965536	Black	Lucy	05/03/58	Engineering

- There is no pair of tuples with the same values on both **`Surname`** and **`DegreeProg`**;

  i.e., in each program students have different surnames; can we conclude that **`Surname`** and **`DegreeProg`** form a key for this relation?

  No! There ***could be*** students with the same surname in the same program

Existence of Keys

- Relations are sets; therefore each relation is composed of <u>distinct</u> tuples.
- It follows that the whole set of attributes for a relation defines a **superkey**.
- Therefore **each relation has a key**, which is the set of all its attributes, or a subset thereof.
- The existence of keys guarantees that each piece of data in the database can be accessed,
- Keys are a major feature of the Relational Model and allow us to say that it is "**value-based**".