The Relational Model

Fall 2017, Lecture 2

> A relationship, I think, is like a shark, you know? It has to constantly move forward or it dies. And I think what we got on our hands is a dead shark.
>
> Woody Allen (from Annie Hall, 1979)

So…*What* Is a Database?

- Database:
  An (often) large, integrated collection of data.

- Models a real-world *enterprise*
  - Entities (e.g., teams, games)
  - Relationships
    (e.g., Cal *plays against* Stanford *in* The Big Game)
  - Can also include active components , often called "business logic". (e.g., the BCS ranking system)

Key Concept: Structured Data

- 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.
  - Every relation has a *schema*, which describes the columns, or fields.

Example: University Database

- Conceptual schema:

  Students(sid*: string*, name*: string*, age*: integer*, gpa*:real)*
  Courses(cid*: string*, cname*:string*, credits*:integer)*
  Enrolled(sid*:string*, cid*:string*, grade*:string)*

  ```
          FOREIGN KEY sid REFERENCES Students
          FOREIGN KEY cid REFERENCES Courses
  ```

- External Schema (View):

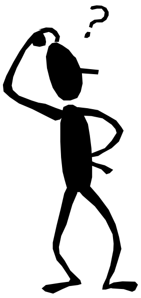
  Course_info(cid:string,enrollment:integer*)*

  ```
  Create View Course_info AS
  SELECT cid, Count (*) as enrollment
  FROM Enrolled
  GROUP BY cid
  ```

e.g.: An Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

What is a Database System?



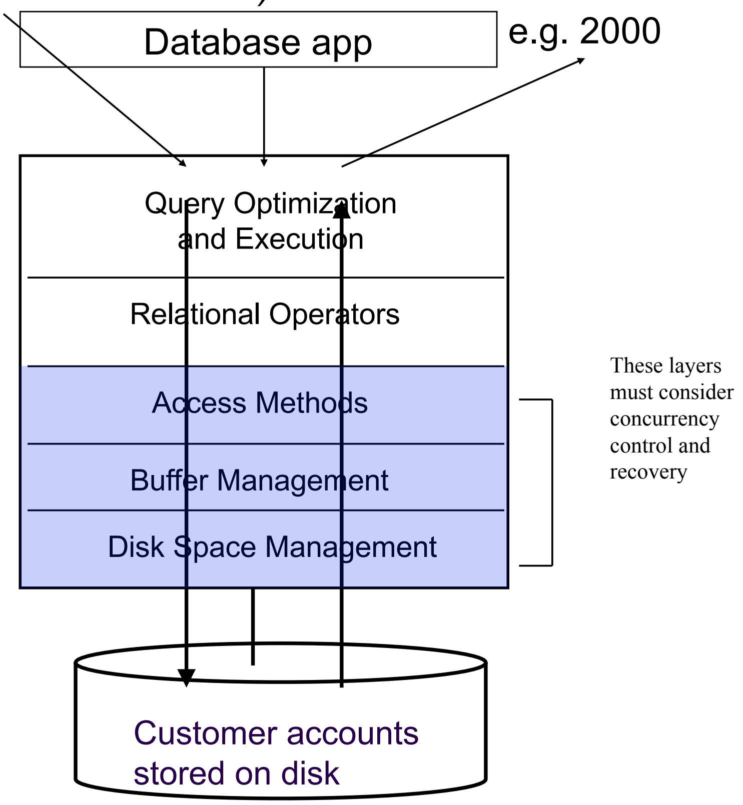
- A *Database Management System (DBMS)* is a software system designed to store, manage, and facilitate access to databases.

- A DBMS provides:
  - Data Definition Language (DDL)
  - Data Manipulation Language (DML)
    - Queries – to retrieve, analyze and modify data.
    - Sometimes called “CRUD”
  - Guarantees about durability, concurrency, semantics, etc.
- Three main uses: Transactional, Archival, and Analytical

A DBMS “Lasagna” Diagram

Query in:
e.g. “*Select min(account balance)*”

Data out:
e.g. 2000

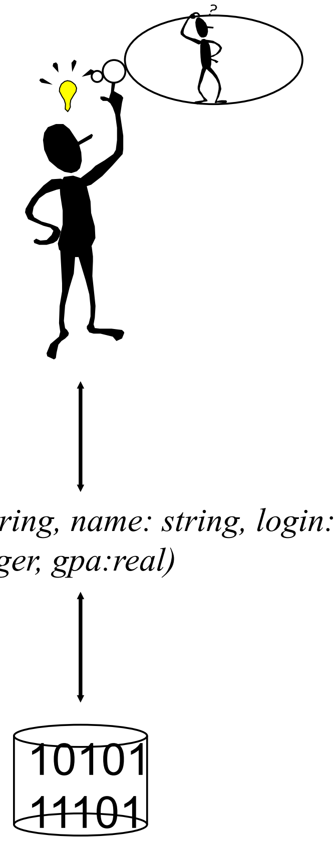


These layers must consider concurrency control and recovery

- The book shows a somewhat more detailed version.

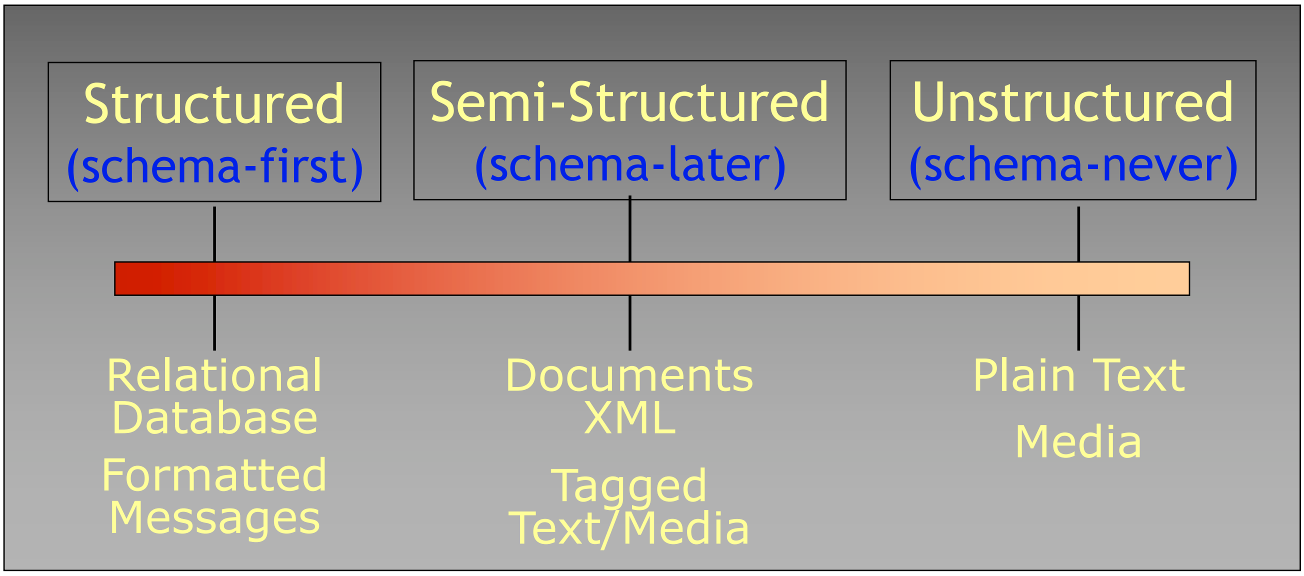
Data Models – Describing Data

- A *Database design* encodes some portion of the real world.
- A *Data Model* is a set of concepts for thinking about this encoding.
- Many models have been proposed.
- We will concentrate on two related models:
  i) Entity-Relationship (graphical)
  ii) Relational (implementation)



Student *(sid: string, name: string, login: string, age: integer, gpa:real)*

The Structure Spectrum



Structured (schema-first)

Semi-Structured (schema-later)

Unstructured (schema-never)

Relational Database

Formatted Messages

Documents XML

Tagged Text/Media

Plain Text

Media

BIG IDEA: “Data Independence”

- First introduced by Codd in 1970
  - *“A relational model of data for large shared data banks”*
  - Recognized as a landmark paper

- [The Relational Model] provides a basis for a high level data language which will yield maximal independence between programs on the one hand and machine representation on the other.

(E.F. Codd, *CACM* 1970)

In Other Words…

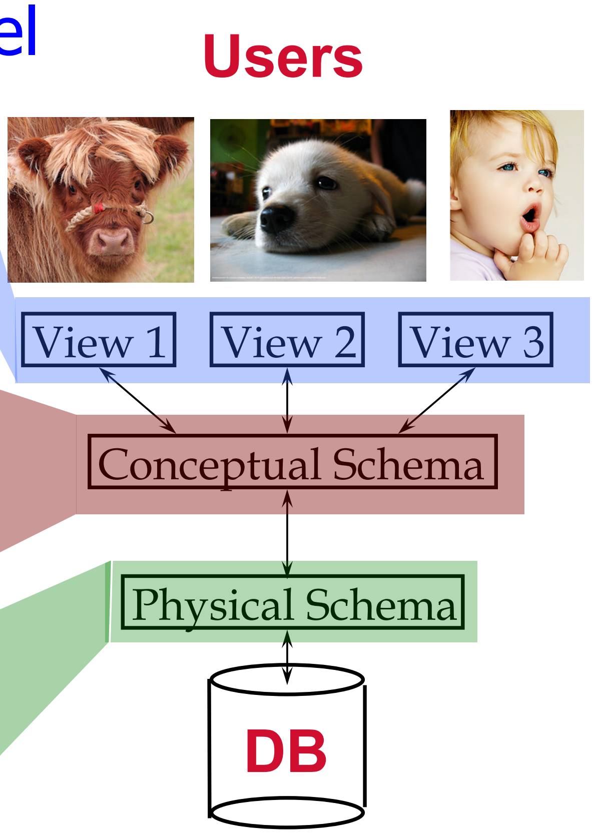
Relational DataBase Management Systems were invented to let you use one set of data in multiple ways, including ways that are unforeseen at the time the database is built and the 1st applications are written.

(Curt Monash, analyst/blogger)

That is, think about the data independently of any particular program.

ANSI/SPARC Model

- Views describe how users see the data.
- Conceptual schema defines logical structure
- Physical schema describes the files and indexes used.



Example: University Database

- Conceptual schema:

  -
    ```
    Students(sid text, name text,
             login text, age integer,
             gpa float)
    ```

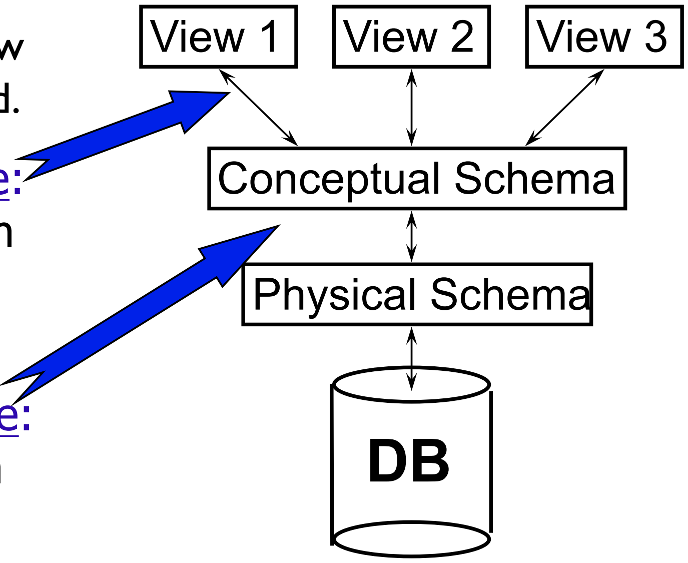
  -
    ```
    Courses(cid text, cname text,
            credits integer)
    ```

  -
    ```
    Enrolled(sid text, cid text,
             grade text)
    ```
- Physical schema:
  - Relations stored as unordered files.
  - Index on first column of Students.
- External Schema (View):

  -
    ```
    Course_info(cid text,
                total_enrollment integer)
    ```

Data Independence: Two Flavors

- A Simple Idea: Applications should be insulated from how data is structured and stored.
- Logical data independence: Protection from changes in *logical* structure of data.
- Physical data independence: Protection from changes in *physical* structure of data.



- Q: Why is this particularly important for DBMS? (compared to your favorite programming language)

Steps in Database Design

- Requirements Analysis
  - user needs; what must the database capture?
- Conceptual Design
  - high level description (often done w/ER model)
- Logical Design
  - translate ER into DBMS data model
    - Typically: “relational” model as implemented by SQL
- Schema Refinement - consistency, normalization
- Physical Design - indexes, disk layout
- Security Design - who accesses what, and how

Implementation: The Relational Model

- Fairly easy to map an E-R design to a Relational Schema
- The Relational Model is Ubiquitous
  - MySQL, PostgreSQL, Oracle, DB2, SQLServer, …
- Object-oriented concepts have been merged in
  - Early work: POSTGRES research project at Berkeley
  - Informix, IBM DB2, Oracle 8i
- As has support for XML (semi-structured data)

Relational Database: Definitions

- *Relational database*: a set of *relations*
- *Relation:* made up of 2 parts:

  Schema : specifies name of relation, plus name and type of each column

  Students(*sid*: string, *name*: string, *login*: string, *age*: integer, *gpa*: real)

  Instance : the actual data at a given time

  - #rows = *cardinality*
  - #fields = *degree / arity*

Some Synonyms

Formal	Not-so-formal 1	Not-so-formal 2
Relation	Table	
Tuple	Row	Record
Attribute	Column	Field
Domain	Type	

Ex: Instance of Students Relation

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

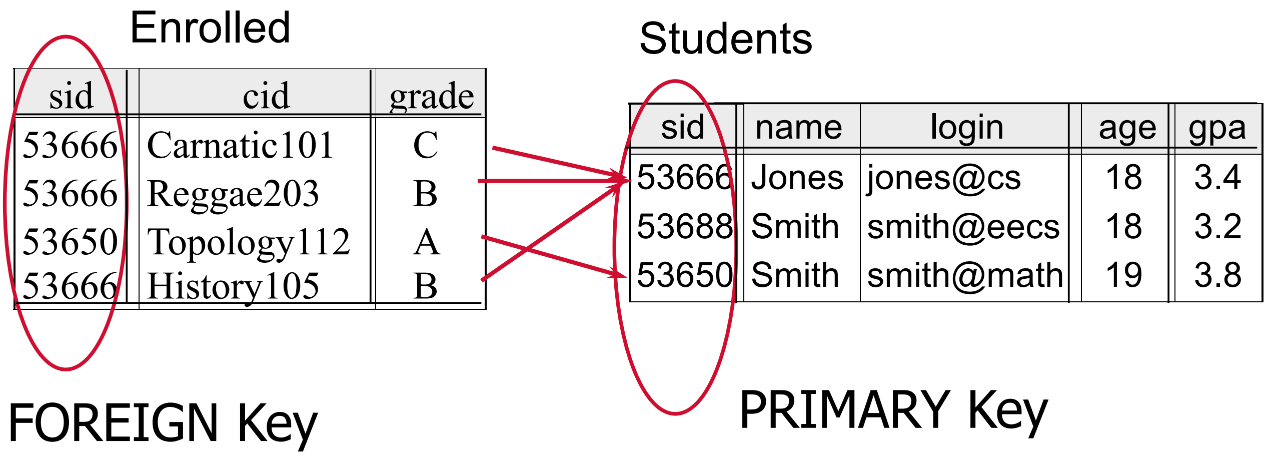
- Cardinality = 3, arity = 5 , all rows distinct
- Do all values in each column of a relation instance have to be distinct?

Keys

- A set of fields is a superkey if:
  - No two distinct tuples can have same values in all key fields
- A set of fields is a key for a relation if :
  - It is a superkey
  - No subset of the fields is a superkey
- what if >1 key for a relation?
  - One of the keys is chosen to be the primary key while other keys are called candidate keys.
- E.g.
  - sid is a key for Students.
  - What about name?
  - The set {sid, gpa} is a superkey.

Keys (Continued)

- Keys are a way to associate tuples in different relations
- Keys are one form of integrity constraint (IC)



Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

FOREIGN Key

PRIMARY Key

SQL Data Languages

- The query and update commands together form the Data Manipulation Language (DML) part of SQL:
  - SELECT - extracts data from a database table
  - UPDATE - updates data in a database table
  - DELETE - deletes data from a database table
  - INSERT INTO - inserts new data into a database table
- The Data Definition Language (DDL) part of SQL permits database tables to be created or deleted:
  - CREATE TABLE - creates a new database table
  - ALTER TABLE - alters (changes) a database table
  - DROP TABLE - deletes a database table
  - CREATE INDEX - creates an index (search key)
  - DROP INDEX - deletes an index

SQL - A language for Relational DBs

- Say: “ess-cue-ell” or “sequel”
- SQL deviates from the pure (set-oriented) relational model, e.g.:
  - can have duplicates, ordering,
- Data Definition Language (DDL)
  - create, modify, delete relations
  - specify constraints
  - administer users, security, etc.
- Data Manipulation Language (DML)
  - Specify retrieval queries
  - add, modify, remove tuples

SQL Overview

- CREATE TABLE <name> (<field> <domain>, …)
- INSERT INTO <name> (<field names>)
  VALUES (<field values>)
- DELETE FROM <name>
  WHERE <condition>
- UPDATE <name>
  SET <field name> = <value>
  WHERE <condition>
- SELECT <fields>
  FROM <name>
  WHERE <condition>

Creating Relations in SQL

- Creates the Students relation.
  - Note: the type (domain) of each field is specified, and enforced by the DBMS whenever tuples are added or modified.

```
CREATE TABLE Students
     (sid CHAR(20),
      name CHAR(20),
      login CHAR(10),
      age INTEGER,
      gpa FLOAT)
```

Table Creation (continued)

- Another example: the Enrolled table holds information about courses students take.

```
CREATE TABLE Enrolled
     (sid CHAR(20),
      cid CHAR(20),
      grade CHAR(2))
```

Adding and Deleting Tuples

- Can insert a single tuple using:

```
INSERT INTO Students (sid, name, login, age, gpa)
 VALUES ('53688', 'Smith', 'smith@ee', 18, 3.2)
```

- **Can delete all tuples satisfying some condition (e.g., name = Smith):**

```
DELETE
  FROM Students S
 WHERE S.name = 'Smith'
```

Powerful variants of these commands are available; more later!

Primary and Candidate Keys in SQL

- Possibly many *candidate keys* (specified using UNIQUE), one of which is chosen as the *primary key*.

- Keys must be used carefully! Say you want to enforce:
  "For a given student and course, there is a single grade."

```
CREATE TABLE Enrolled
  (sid CHAR(20)
   cid  CHAR(20),
   grade CHAR(2),
   PRIMARY KEY (sid,cid))
```

vs.

```
~~CREATE TABLE Enrolled
   (sid CHAR(20)
    cid  CHAR(20),
    grade CHAR(2),
    PRIMARY KEY  (sid),
    UNIQUE (cid, grade))~~
```

Right hand schema: "Students can take only one course, and no two students in a course receive the same grade."

Foreign Keys, Referential Integrity

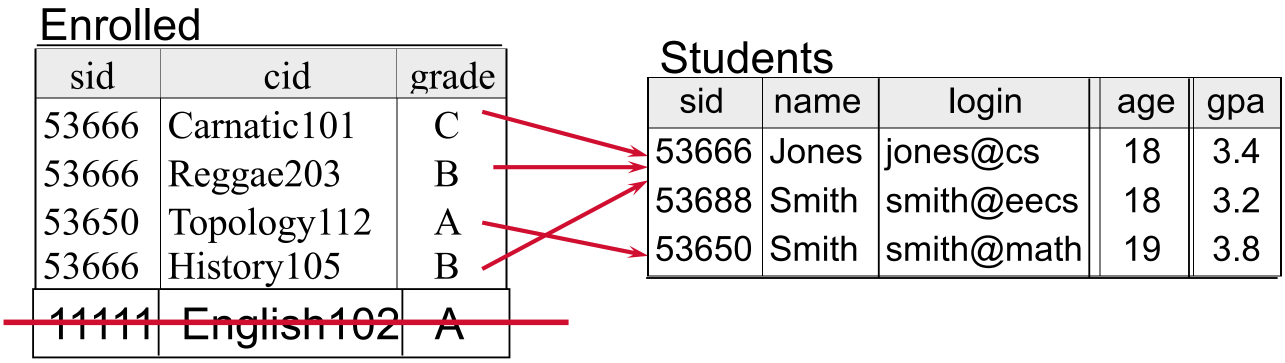
- *Foreign key*: a “logical pointer”
  - Set of fields in a tuple in one relation that `refer’ to a tuple in another relation.
  - Reference to *primary key* of the other relation.

- All foreign key constraints enforced?
  - *referential integrity*!
  - i.e., no dangling references.

Foreign Keys in SQL

- **E.g. Only students listed in the Students relation should be allowed to enroll in courses.**
  - *sid* is a foreign key referring to Students:

```
CREATE TABLE Enrolled
(sid CHAR(20),cid CHAR(20),grade CHAR(2),
 PRIMARY KEY (sid,cid),
 FOREIGN KEY (sid) REFERENCES Students);
```



Enrolled

sid	cid	grade
53666	Carnatic101	C
53666	Reggae203	B
53650	Topology112	A
53666	History105	B
~~11111~~	~~English102~~	~~A~~

Students

sid	name	login	age	gpa
53666	Jones	jones@cs	18	3.4
53688	Smith	smith@eecs	18	3.2
53650	Smith	smith@math	19	3.8

Enforcing Referential Integrity

- *sid* in Enrolled: foreign key referencing Students.
- Scenarios:
  - Insert Enrolled tuple with non-existent student id?
  - Delete a Students tuple?
    - Also delete Enrolled tuples that refer to it? (Cascade)
    - Disallow if referred to? (No Action)
    - Set sid in referring Enrolled tuples to a *default* value? (Set Default)
    - Set sid in referring Enrolled tuples to *null*, denoting `*unknown*' or `*inapplicable*'. (Set NULL)
- Similar issues arise if primary key of Students tuple is updated.

Steps in Database Design

- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - high level description (Ex. ER model)
- Logical Design
  - translate ER into DBMS data model
- Schema Refinement
  - consistency, normalization
- Physical Design
  - indexes, disk layout
- Security Design
  - who accesses what, and how