Foundation of Relational Databases

Yanlei Diao

Slides Courtesy of R. Ramakrishnan and J. Gehrke

Teaching Staff

- Instructor: Yanlei Diao
  - Email: yanlei.diao@polytechniqu.edu
  - How to address in email: “Dear Prof. Diao”, “Dear Yanlei”

- Teaching Assistants
  - Michaël Thomazo michael.thomazo@inria.fr
  - Oscar santiago Mendoza rivera oscar-santiago.mendoza-rivera@inria.fr
  - Prosper burq prosper.burq@gmail.com

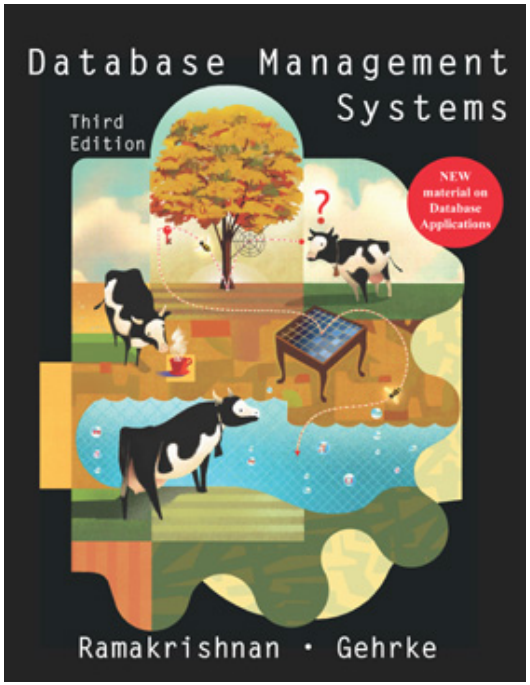
Course Web Site

http://datascience-x-master-paris-saclay.fr/

Follow the link to **Moodle:**

- **post lecture notes**
- **collect submissions**

Textbook



Database Management Systems 3rd Edition

Ramakrishnan and Gehrke

Amazon:

- Buy new: $43-$147.09 (hardcover); paperback, $23; Kindle, rent options are also available…

Lecture notes will be posted in **Moodle** after class.

Academic Honesty

- All submitted work must be your own!
  - Although students are encouraged to study together, each student much produce his or her own solution to each homework.
  - Copying or using **sections of someone else's program or assignment** (even if it has been modified by you), or copying a **solution from an external source**, is not acceptable**.**
  - The teaching staff will be vigorous in enforcing them.

Databases and DBMS's

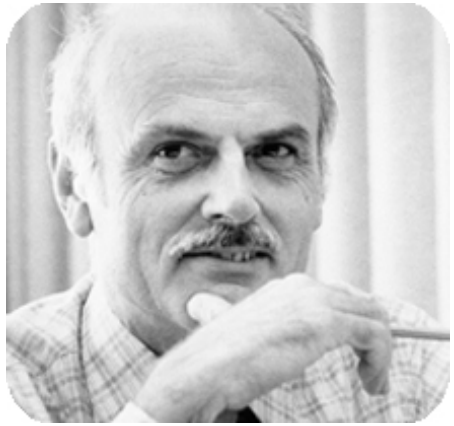
- A **database** is a large, integrated collection of data
- A **database management system** (DBMS) is a software system designed to store and manage a large amount of data
  - Declarative interface to define data stored, add data, update data, and query data
  - Efficient querying
  - Concurrent users
  - Reliable storage and crash recovery
  - Access control…

Early DBMS's

- Early DBMS's (1960's) evolved from file systems
- Many small data items, many queries and updates
  - Banking, Airline reservations
- 1960s Navigational DBMS
  - Tree-based or graph-based data model
  - Manual navigation to find what you want
  - No support for "search" (≠ "program")
- 1973 Turing Award Winner
  - Charles William Bachman
  - "The Programmer as Navigator"
  - The network data model

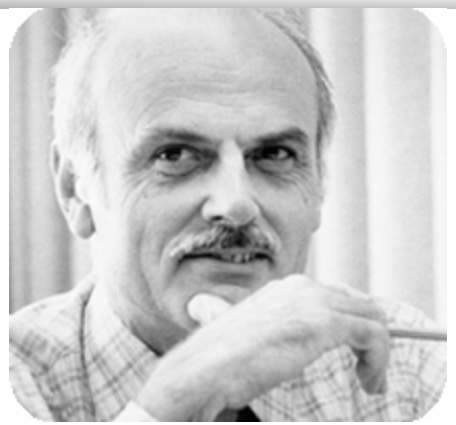
Relational DBMS

- Relational model (1970)
  - Data independence: hides details of physical storage from users
  - Declarative query language: say **what** you want, not **how** to compute it
  - Mathematical foundation: what queries mean, possible implementations
- 1981 Turing Award Winner
  - Edgar F. ("Ted") Codd
  - Mathematically-inclined researcher
  - Legitimized DBMSs as a theoretically respectable research field in CS



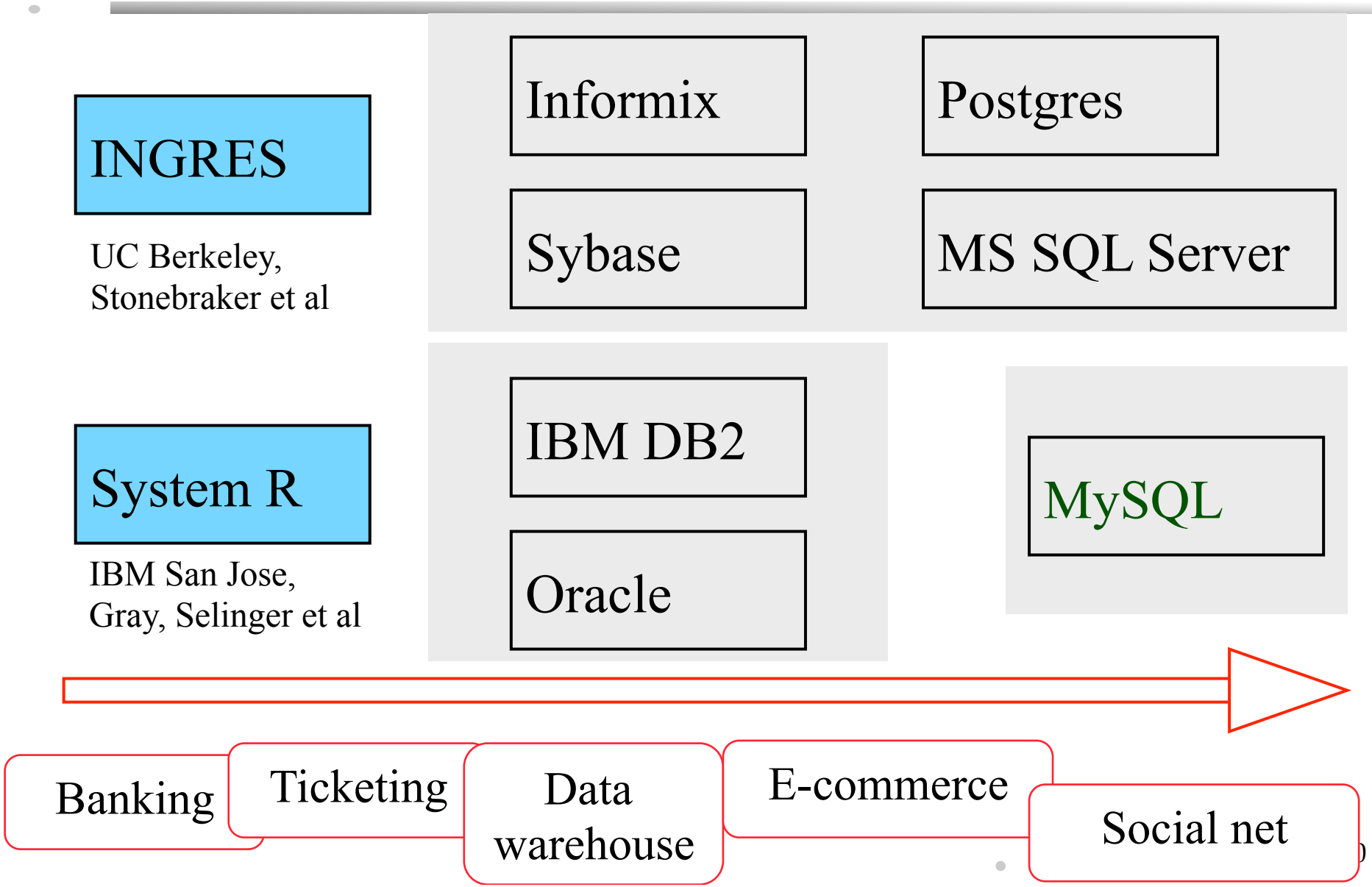
Relational DBMS

1974 Debate at an ACM Workshop



- Query optimization (1970's till now)
  - Earliest: System R at IBM, INGRES at UC Berkeley
  - Queries can be efficiently executed despite data independence and declarative queries!
- 2014 Turing Award Winner
  - Michael Stonebraker
  - "For fundamental contributions to modern database systems"

Commercial DBMS's



Foundation of Relational Databases

Foundation of Relational Databases

- Relational Model
- Formal Query Languages
  - Relational Algebra
  - Relational Calculus
  - Language Theory

Relational Model

- A relational database is a set of *relations*.
- Each relation has:
  - *Schema* : specifies name of relation, name and type (domain) of each attribute.
    - Students(*sid*:string, *name*:string, *login*:string, *age*:integer, *gpa*:real).
  - *Instance* : a table with rows (*tuples*) and columns (*attributes, fields*).

  cardinality = #rows, *degree / arity* = #columns.
- A relation is a *set* of tuples (in theory).
  - All rows must be distinct, no duplicates.

Example 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, degree = 5
- All rows are distinct.
- Some columns of two rows can be the same.

Creating Relations in SQL

- Create the Students relation

- Specify *domain constraints*:
  - type of each field
  - later enforced by the DBMS upon tuple insertion or update.

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

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

Adding Tuples

- Can insert a single tuple using:

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

☛ *Powerful variants of these commands are available; more later!*

Integrity Constraints

- *Integrity Constraints* (IC's): condition that must be true for **any** instance of the database.
  - *Domain constraint*
  - *Primary key constraint*
  - *Foreign key constraint*
  - Specified when the schema is defined.
- DBMS enforces ICs.
  - Stored data is faithful to real-world meaning.
  - Avoids data entry errors, too!

Primary Key Constraints

- *Key* of a relation: minimal set of attributes that uniquely identify each entity.
  1. No two tuples can have same values in all key fields.
  2. This is not true for any subset of the key.
  - Part 2 false? A *superkey*.
  - If more than 1 key for a relation, *candidate keys*.
  - One of candidate keys is chosen to be the *primary key*.
- E.g., Students(sid, name, login, age, gpa)

Primary and Candidate Keys in SQL

- Specify *candidate keys* using UNIQUE.
- Choose one candidate key as the *primary key*.

"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));
```

"… and no two students in a course receive the same grade."

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

Foreign Keys

- *Foreign key*: set of fields used to `refer' to the primary key of another relation.
  - Like a `logical pointer'.
- E.g., Enrolled(*sid*: string, *cid*: string, *grade*: string):
  - *sid* is a foreign key referring to Students.

Foreign Keys in SQL

- Only students listed in the Students relation should be allowed to enroll for courses.

```
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

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

Referential Integrity

- *Referential integrity*: any foreign key value must have a matching primary key value in referenced reln.
  - E.g., every *sid* value in Enrolled must appear in Students.
  - No dangling references.
- Can you name a data model w/o referential integrity?

Enforcing Referential Integrity

- What if an Enrolled tuple with a non-existent student id is inserted?
  - *Reject it!*
- What if a Students tuple is deleted?
  - CASCADE: delete all Enrolled tuples that refer to it.
  - NO ACTION: disallow if the Students tuple is referred to.
  - SET DEFAULT: set the foreign key to a *default sid*.
  - SET NULL: set the foreign key to a special value *null*, denoting *`unknown'* or *`inapplicable'*.
- Updates to *sid* in Students are treated similarly.

Referential Integrity in SQL

```
CREATE TABLE Enrolled
  (sid CHAR(20),
   cid CHAR(20),
   grade CHAR(2),
   PRIMARY KEY  (sid,cid),
   FOREIGN KEY (sid)
     REFERENCES Students
          ON DELETE CASCADE
          ON UPDATE NO ACTION);
```

Where do IC's Come From?

- Based upon real-world business logic.
- Can check violation in a database instance, but can NEVER infer an IC by looking at an instance.
  - An IC is a statement about *all possible* instances!
  - E.g., *name* of the Students relation.

Questions