Introduction

Ramin Moazzeni

San Jose State University

CS157a – Database Mgmt Systems

CS 157a: Database Mgmt Systems

Instructor: Ramin Moazzeni, Ramin.Moazeni@sjsu.edu

- Office hours: after class (or by appointment)

Course web site: https://sjsu.instructure.com

Texts and readings:

- Ramakrishnan & Gerke, *Database Management Systems*, 3rd ed.
- For SQL: Greenspun, *SQL for Nerds* (http://philip.greenspun.com/sql/index.html)
- Other books may be useful:
  - Elmasri, Navathe. "Fundamentals of Database Systems," Latest Edition
- Oracle References
  - Oracle SQL Reference
  - Oracle SQL*Plus User's Guide and Reference
  - Online Oracle Documentations
- Various papers and readings
- Prerequisites:
  - CS 146 (with a grade of "C-" or better); Computer Science, Applied and Computational Math, Forensic Science: Digital Evidence, or Software Engineering majors only; or instructor consent.

Course Format and Grading

- Quizzes: 10%
- HW Assignments: 20% (Individual)
- Project: 20% (Group)
  - Specification will be provided later
- Midterm exam: 25% (Individual)
- Final exam: 25% (Individual)

Important Dates

- Class starts Tuesday, June 1
  Midterm Exam: Tuesday, July 6 (11am-1pm) ONLINE
  Class ends Thursday, Aug 5
  Final Exam: Thursday, Aug 5 (11am-1pm) ONLINE

Course Outline (may vary slightly)

Outline

- ER Data model
- The relational data model
- Map ER to Relational model
- Extended ER
- SQL
- OO & ORDBMS
- Application Programming (DB connectivity)
- XML, JSON, XML Schema
- Normalization
- Transactions
- DB Security
- Spatial DBs – If time permits
- NoSQL – If time permits

Cheating/Plagiarism

- Cheating is a serious offense.
- Cheating includes copying on exams or written assignments; obtaining advance copies of exams; outsourcing homeworks or project work; and copying material from the web and including on homeworks without proper attribution.
- You may discuss concepts with your classmates or anyone else, and you are encouraged to do so. However, when it comes to writing the program (even just the 'pseudo-code'), you must do it yourself
- Please see the university's policy regarding academic integrity: http://www.sjsu.edu/senate/docs/S07-2.pdf

Outline for Today's Lecture

- Overview of database systems
  - Recommended reading: Introduction of **SQL for Web Nerds,** by Philip Greenspun http://philip.greenspun.com/sql/
  - Read Chapter 1 of the textbook
- Course outline
- What the course is about

What *Is* a Database Management System ?

- Database: collection of files that store the data
  - Entities: students, faculty, courses and classroom
  - Relationships: between entities
    - Students taking courses
    - Faculty teaching courses
- DataBase Management System = DBMS
  - Software that Manages the DB
  - A big program that accesses and updates those files
- Relational DBMS = RDBMS
  - DBMS that is based on a relational model

Where are RDBMS used ?

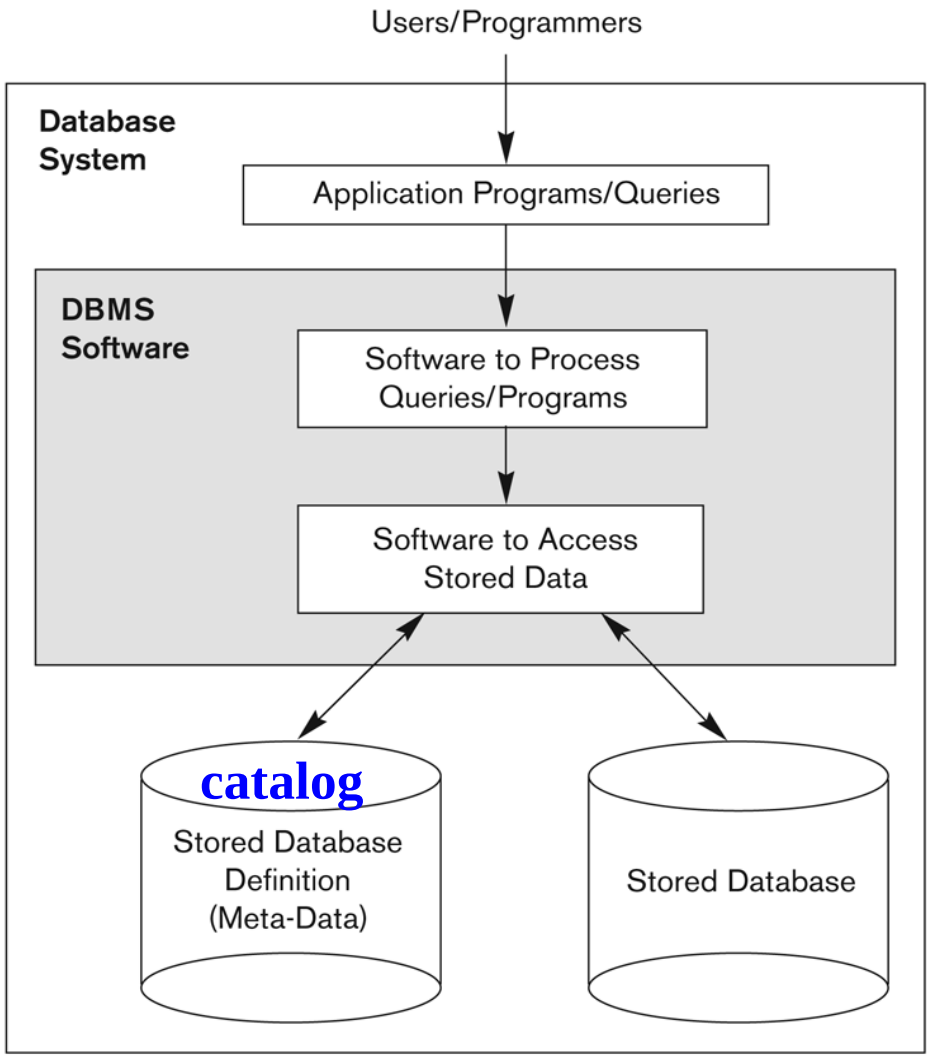
- Backend for traditional “database” applications
- Backend for large Web sites
- Backend for Web services

Examples of DBMS Usage

- Airlines: reservations and schedules (Expedia)
- Universities: student info, grades
- Banking: customer info and accounts
- Credit Cards: customer info, transactions
- Sales: customer info, inventory (Amazon, EBay)
- Government: taxes, census

Example: Internet Movie Database (IMDB)

How is a RDBMS used ?



Databases are self-describing: catalog describes the structure of the data stored in the DB

Example of a Traditional Database Application

- Suppose we are building a system to store information about:
  - students
  - courses
  - professors
  - who takes what, who teaches what

Can we do it without a DBMS ?

Yes…Start by storing the data in files:

students.txt courses.txt professors.txt

Now write C or Java programs to implement specific tasks (i.e store, modify and query)

Doing it without a DBMS…

- Enroll “John Smith” in “COEN444”:

Write a program to do the following:

Read ‘students.txt’
Read ‘courses.txt’
Find&update the record “John Smith”
Find&update the record “COEN444”
Write “students.txt”
Write “courses.txt”

Problems without an DBMS…

- System crashes:

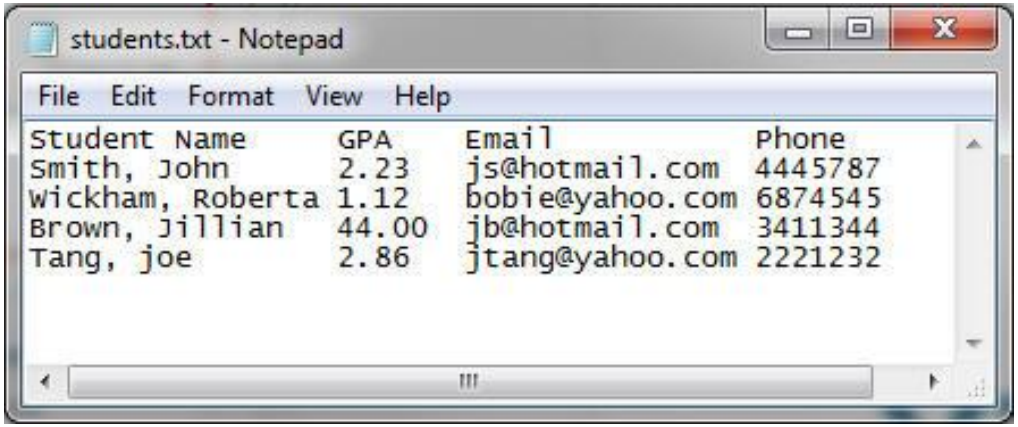
  - What is the problem ?
- Large data sets (say 50TB)
  - What is the problem ?
- Simultaneous access by many users
- Enforcing Constraints
- Scalability
- Security

Enforcing Constraints

- With the text file solution, there is no way to enforce integrity constraints on the data. In other words people can put bad data into the text file.

- In contrast, a DBMS allows us to enforce all kinds of constraints. This really helps (but does not guarantee) that our data is correct.

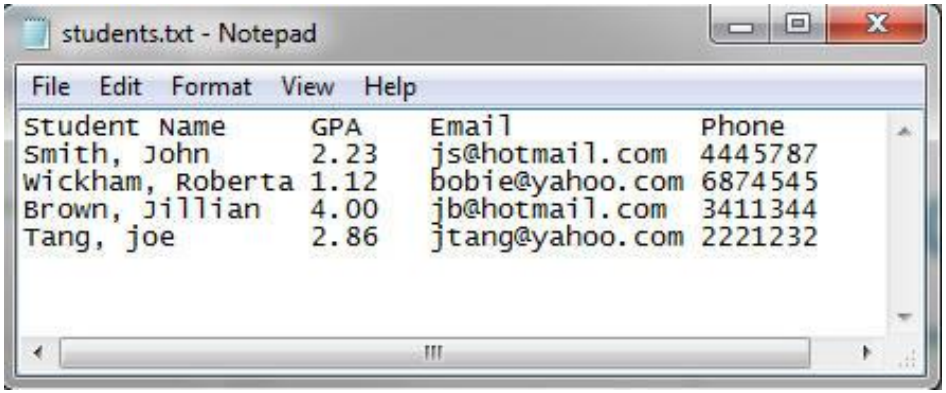
A typo gives Jillian Brown a GPA of 44.00



Scalability

- The text file solution might work for small datasets. What happens when we have big datasets?

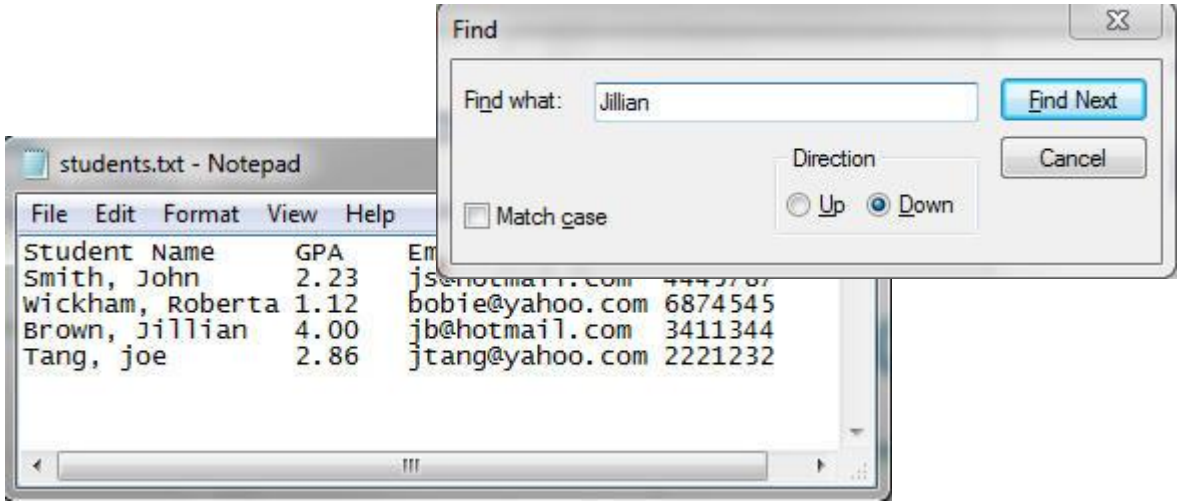
- Most real world datasets are so large that we can only have a small fraction of them in main memory at any time, the rest has to stay on disk.



Query Expressiveness

- The text file solution would allows to search for keywords or certain numbers (slowly).

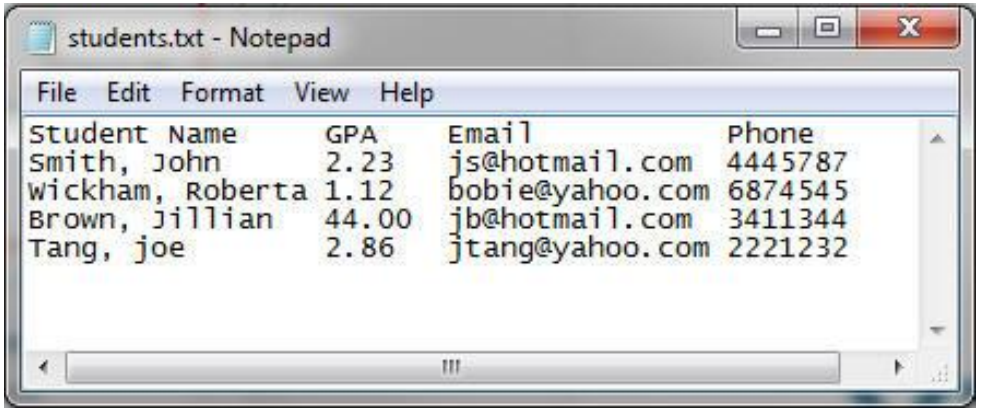
- With a DBMS I can search with much more expressive queries. For example I can ask.. "*Find all students whose GPA is greater than 2.5, and who don't own a phone*" or "*what is the average GPA of the students*"



Query Expressiveness II

- Could write some program that might allow more expressive queries on the text file, but it would be tied into the structure of my data and the operating system etc..

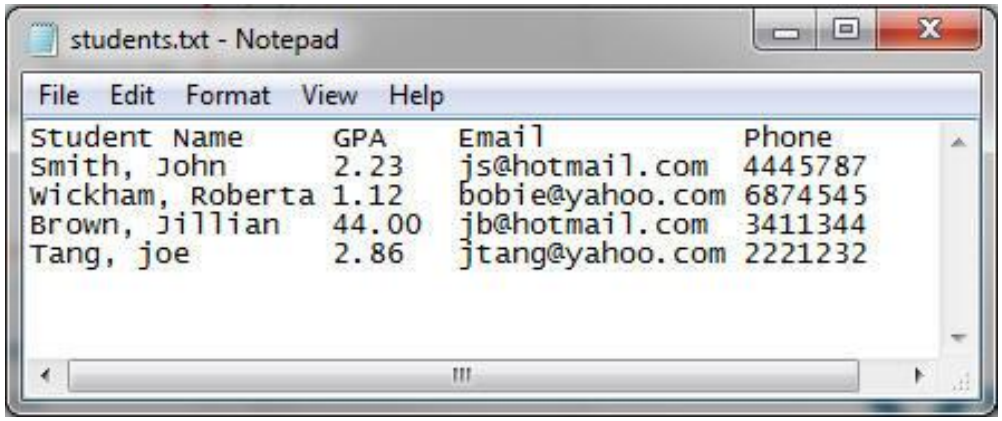
- With a DBMS we are completely isolated from the physical structure of our data. If we change the structure of our data (by adding a field, for example) or moving from a PC to a Mac, nothing changes at the front end!



Different Views

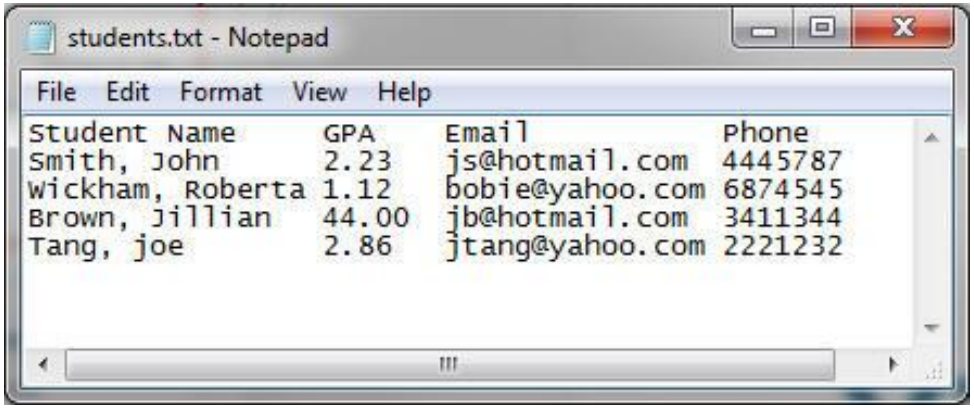
- The text file solution only allows one view of the data.

- With a DBMS, could arrange for different people to have different views of the data. For example, a professor can see everything, while student can see only his/her data, and a TA can see data for students in his/her section, etc.



Concurrency

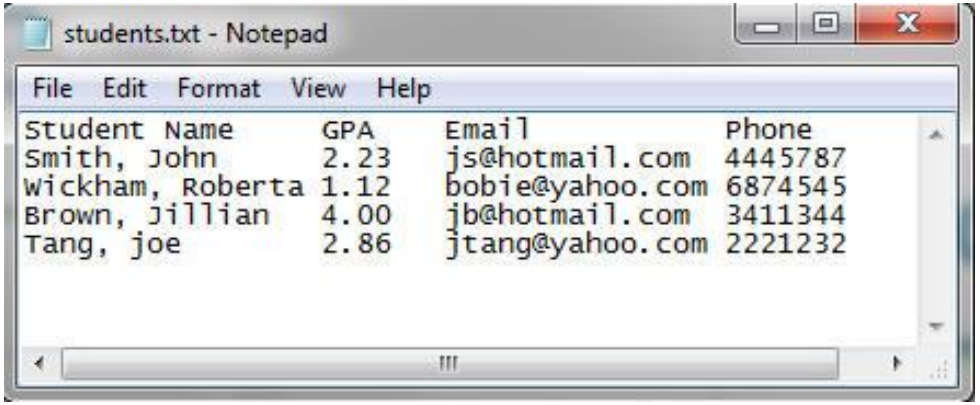
- Suppose the text file is being modified at the same time by multiple users (i.e professor and the TA)
- A DBMS will automatically make sure that this kind of thing cannot happen.



Security

- Suppose the text file on UNIX account, and a student hacks in and changes their grades…

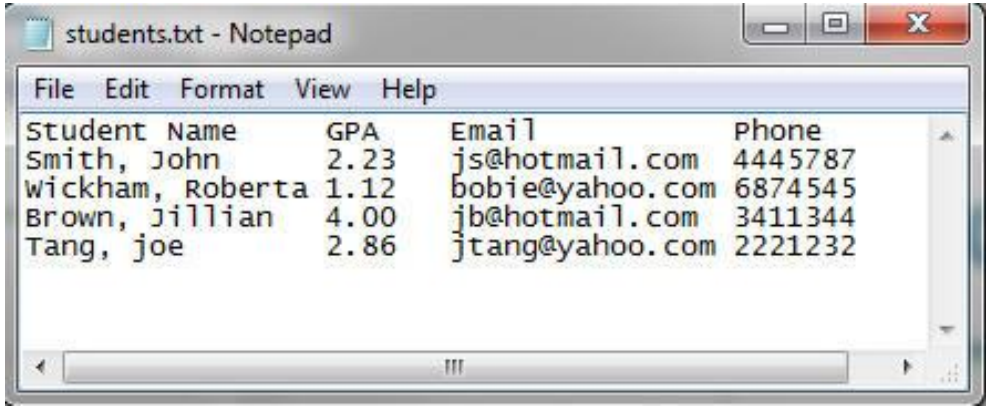
- A DBMS will allow multiple levels of security.
  - Enforce security policies for users to access different subset of data



Crash Recovery

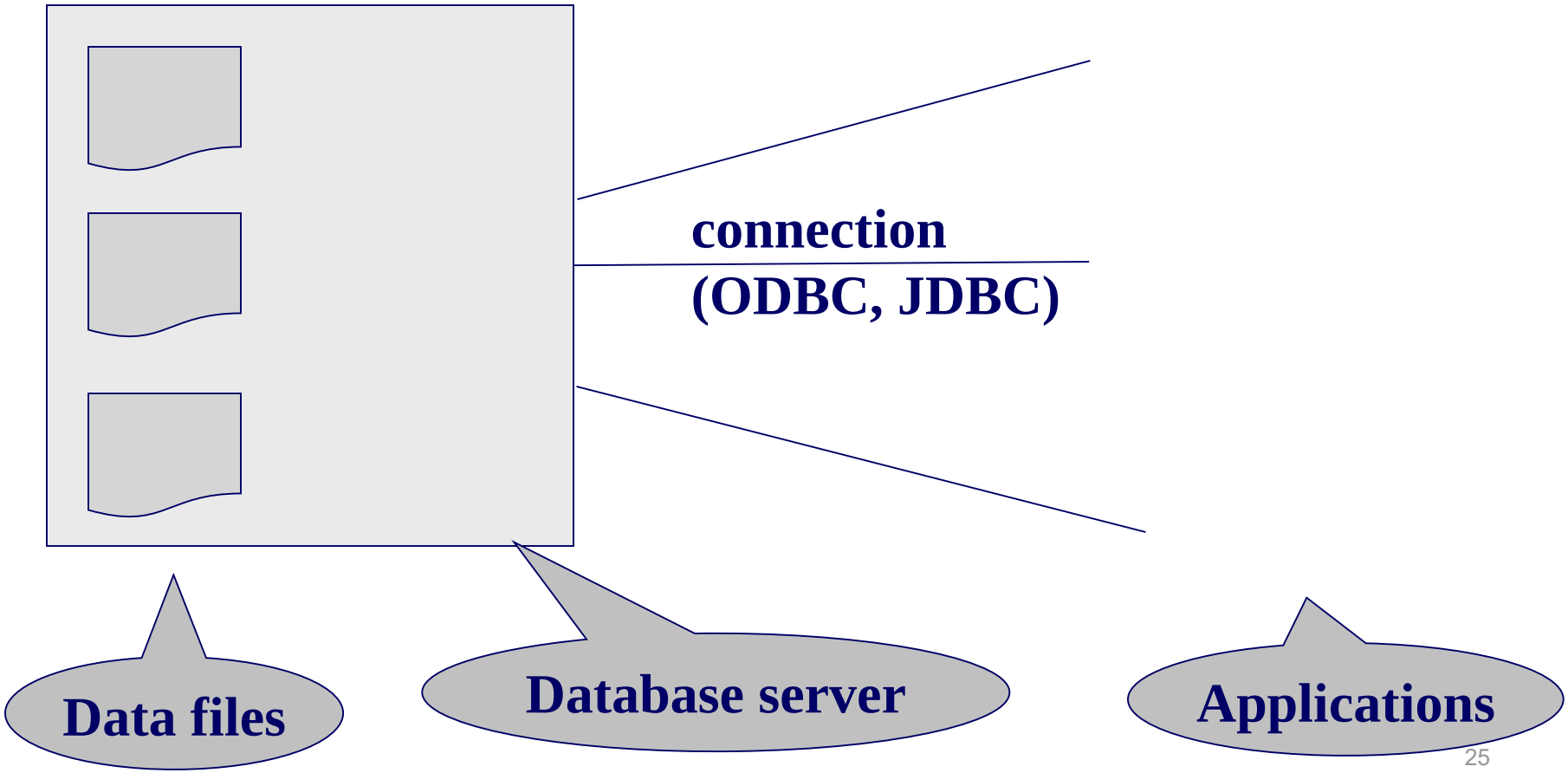
- Suppose while modifying the text file, the system crashes!

- A DBMS is able to guarantee 100% recovery from system crashes (to a consistent state).

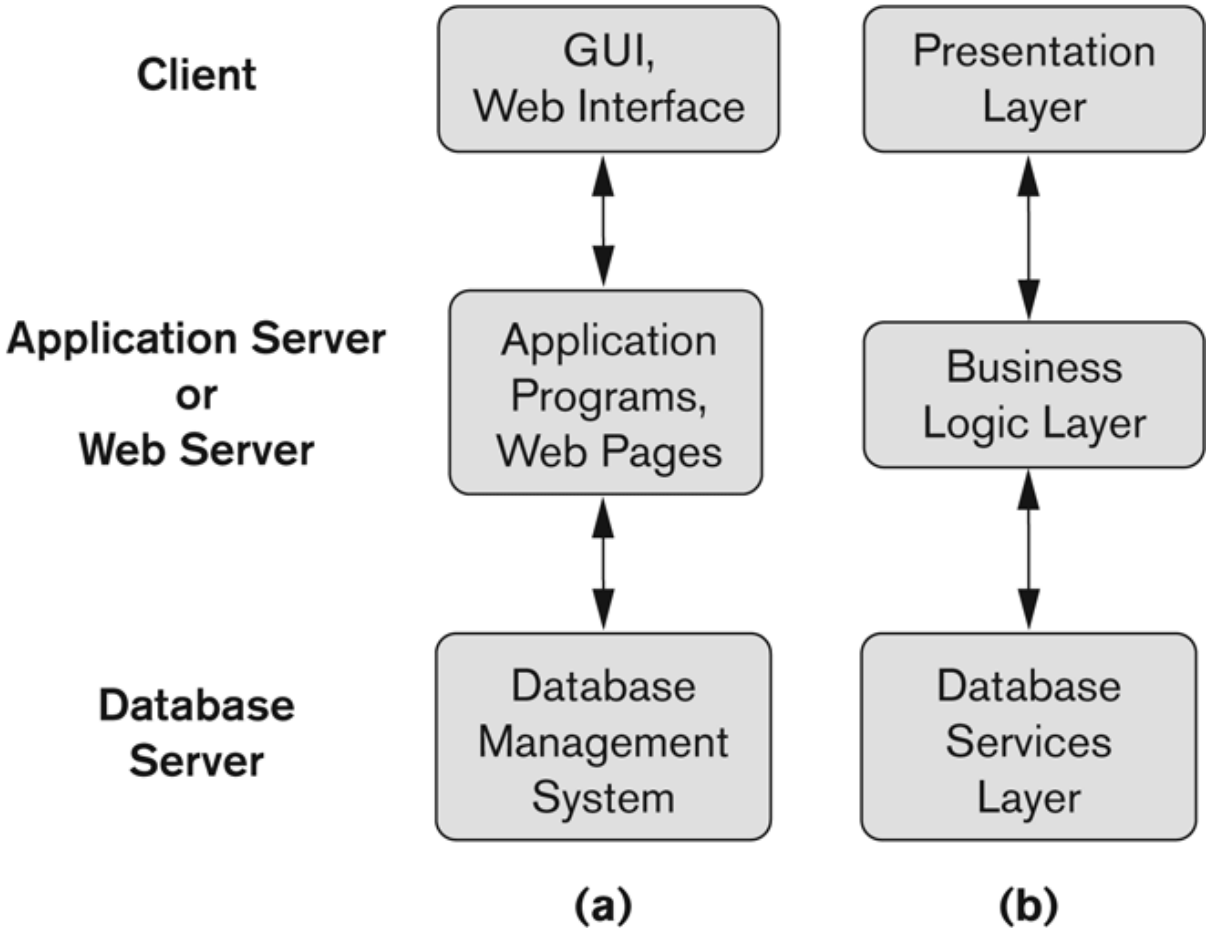


Building the Applications (2-tier)

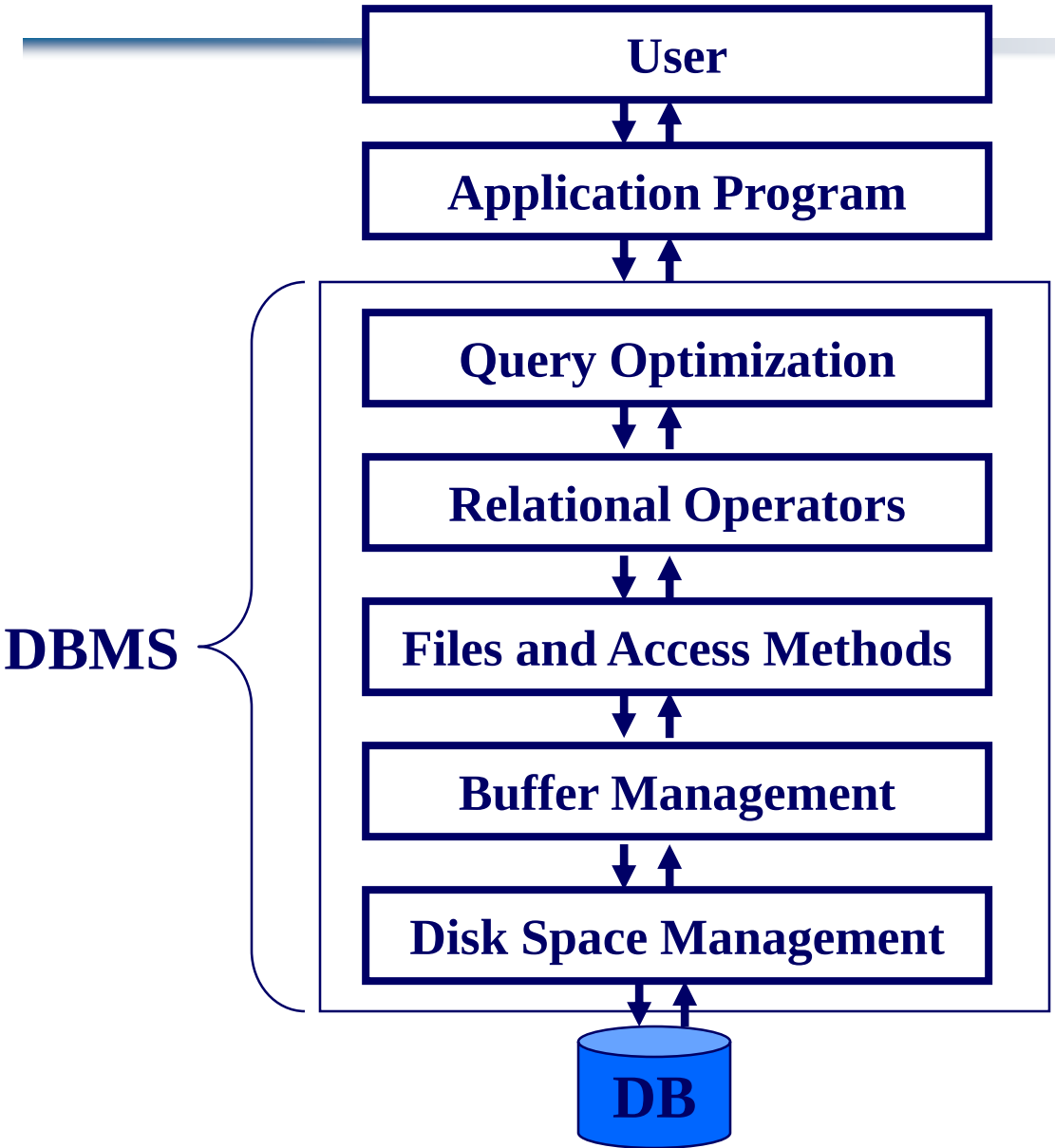
“Two tier database system”



Building the Applications (3-tier)



Control Abstraction



Each layer need not know (or care) how other layers are implemented

The Database Abstraction Provided by the DBMS

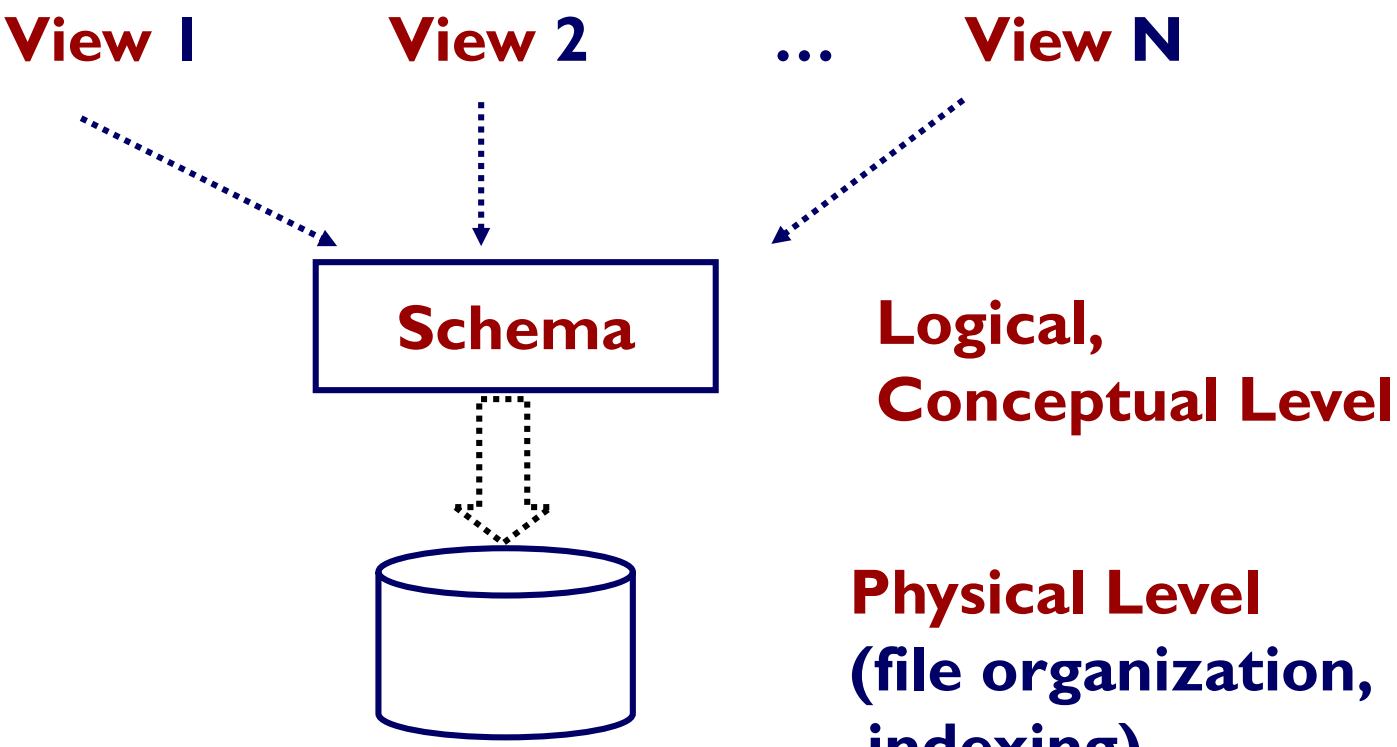
We think of databases at two levels:

- Logical structure:
  - What users/programmers see – program or query interface
- Physical structure:
  - Organization on disk, indices, etc.

The logical level is further split into:

- Overall database design (conceptual; seen by the DB designer)
- Views that various users get to see

The Three-level Architecture for Databases



Functionality of a DBMS

The programmer sees SQL, which has two components:

- Data Definition Language – DDL
  - Creating database tables
- Data Manipulation Language - DML
  - query language

Behind the scenes the DBMS has:

- Query engine
- Query optimizer
- Storage management
- Transaction Management (concurrency, recovery)

How the Programmer Sees the DBMS

- Start with DDL to *create tables*:

```
CREATE TABLE Students (
        Name CHAR(30)
        SSN CHAR(9) PRIMARY KEY NOT NULL,
        Category CHAR(20)
) . . .
```

- Continue with DML to *populate tables:*

```
INSERT INTO Students
VALUES('Charles', '123456789', 'undergraduate')
. . . .
```

How the Programmer Sees the DBMS

- Tables:

Students:

SSN	Name	Category
123-45-6789	Charles	undergrad
234-56-7890	Dan	grad
	…	…

Takes:

SSN	CID
123-45-6789	CSE444
123-45-6789	CSE541
234-56-7890	CSE142
	…

Courses:

CID	Name	Quarter
CSE444	Databases	fall
CSE541	Operating systems	winter

- Still implemented as files, but behind the scenes can be quite complex

"*data independence*" = separate *logical* view from *physical implementation*

Transactions

- Enroll “John Smith” in “COEN444”:

```
BEGIN TRANSACTION;

INSERT INTO Takes
   SELECT Students.SSN, Courses.CID
   FROM Students, Courses
   WHERE Students.name = ‘John Smith’ and
               Courses.name = ‘Databases’

-- More updates here....

IF everything-went-OK
     THEN COMMIT;
ELSE ROLLBACK
```

If system crashes, the transaction is still either committed or aborted

Transactions

- A *transaction* = sequence of statements that either all succeed, or all fail
- Transactions have the ACID properties:

A = atomicity

each transaction be "all or nothing": if one part of the transaction fails, the entire transaction fails

C = consistency

Any transaction will bring the database from one valid state to another

I = independence

ensures that the concurrent execution of transactions results in a system state that would be obtained if transactions were executed serially

D = durability

once a transaction has been committed, it will remain so, even in the event of power loss, crashes, or errors.

Queries

- Find all courses that “John” takes

```
SELECT  C.name
FROM    Students S, Takes T, Courses C
WHERE   S.name=“John” and
            S.ssn = T.ssn and T.cid = C.cid
```

- What happens behind the scene ?
  - Query processor figures out how to answer the query efficiently.

Advantages of a DBMS

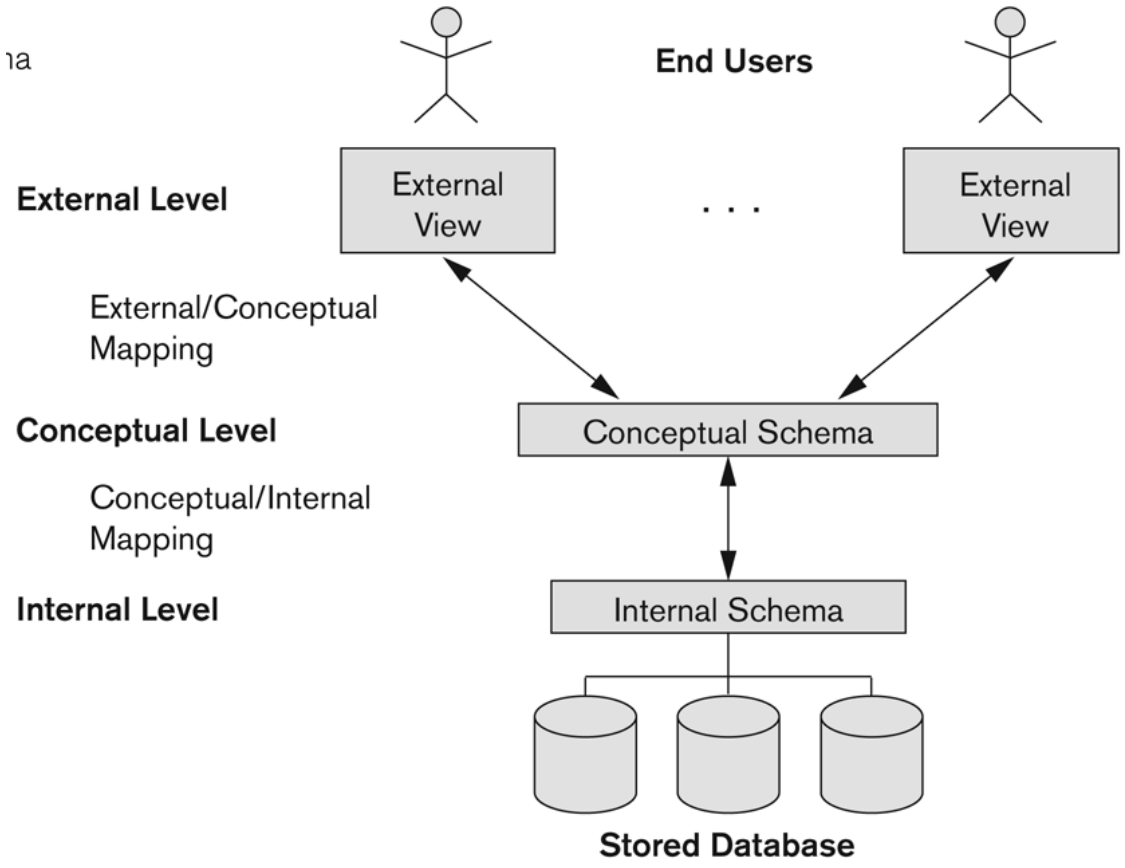
- Data Independence
  - Logical Data Independence

    Ability to change the logical (conceptual) schema without changing the External schema (User View)
  - Physical Data Independence

    Ability to change the physical schema without changing the logical schema

    Protection from changes in physical structure of data.
- Query expressiveness
- Reduced application development time.
- Concurrency and Crash Recovery
  - Schedule concurrent access to the data
- Security
  - Enforce access controls

Data Independence



Each layer need not know how other layers organize data

Data Independence

A user of a relational database system should be able to use the database without knowing about how the precisely how data is stored, e.g.

```
SELECT When, Where
FROM Calendar
WHERE Who = “Jane"
```

After all, you don't worry about the IEEE floating-point specs when you do division in a Java program or with a calculator

More on Data Independence

Logical data independence

Protects the user from changes in the logical structure of the data:

could reorganize the calendar “schema” without changing how we query it

Physical data independence

Protects the user from changes in the physical structure of data:

could add an index on who (or sort by when) without changing how the user would write the query, but the query would execute faster (query optimization)

Why Learn about DBMS?

- Many decisions about how to use a DBMS for an application depend on the capabilities of the DBMS
- To use it well, it's necessary to also understand how a DBMS *works*.

Database Users

- End users (or DB application users)
- DB application programmers (more precisely, they are *DBMS* users)
  - E.g. webmasters
- *Database administrator (DBA)*
  - Designs logical /physical schemas
  - Handles security and authorization
  - Data availability, crash recovery
  - Database tuning as needs evolve

Must understand how a DBMS works!

Data Model

- Data Model:
  - Collection of concepts for describing data
- Schema
  - Description of a particular collection of data, using the a given data model.
- Relational model of data
  - Main concept: relation, basically a table with rows and columns.
  - Every relation has a schema, which describes the columns, or fields

ER Model Basics

- Entity:
  - A real world object or thing
- Attribute
  - Each entity has attributes
- Entity Set
  - Collection of similar entities
  - All entities in an entity set have the same set of attributes

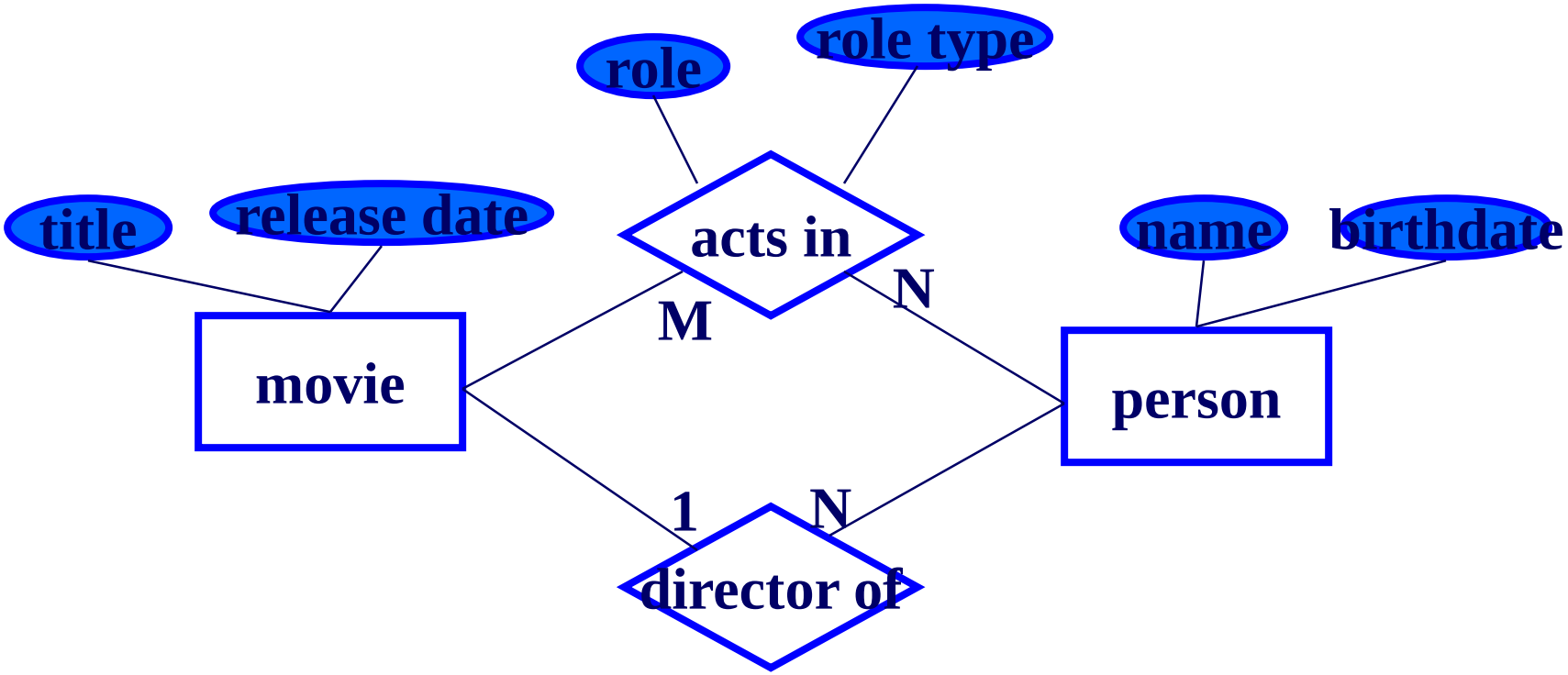
An entity can be uniquely identified thru its attributes

Overview of Database Design

- *Conceptual design*
  - Use *ER Model:* E- *Entities* and R-*Relationships*
  - Decide the *entities* and *relationships* in the enterprise.
  - Decide what information about these entities and relationships should we store in the database.
  - Decide the *integrity constraints* or *business rules.*
- *Implementation (logical design)*
  - Map an ER model into a relational schema.

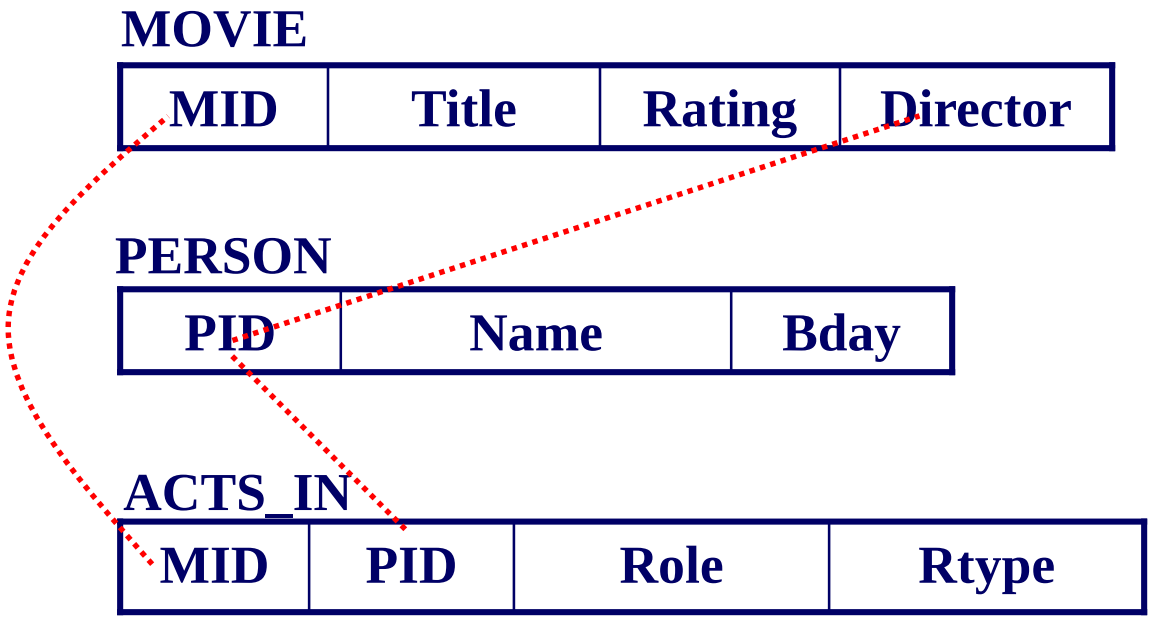
Building a DB: construct a conceptual model

- A conceptual model identifies entities and relationships



Building a DB: Define Relational *Schema*

- A schema describes DB using *data model* supported by DMBS (eg, *relational model)*
- RDBMS – DBMS that supports relational model



Building a DB: Populate DB

MOVIE

MID	Title	Rating	Director
1	The Big Lebowski	R	72
2	Star Wars	PG	29
...			

ACTS_IN

MID	PID	Role	Rtype
1	1	The Dude	STAR
2	2	Han Solo	CO_STAR
...			

PERSON

PID	Name	Bday
1	Jeff Daniels	12/4/49
2	Harrison Ford	7/13/42
...		

Set initial records of the DB

Querying The Database

- Most RDBMS allow users to query the database using SQL (structured query language)
- Example: get cast of “The Big Lebowski”

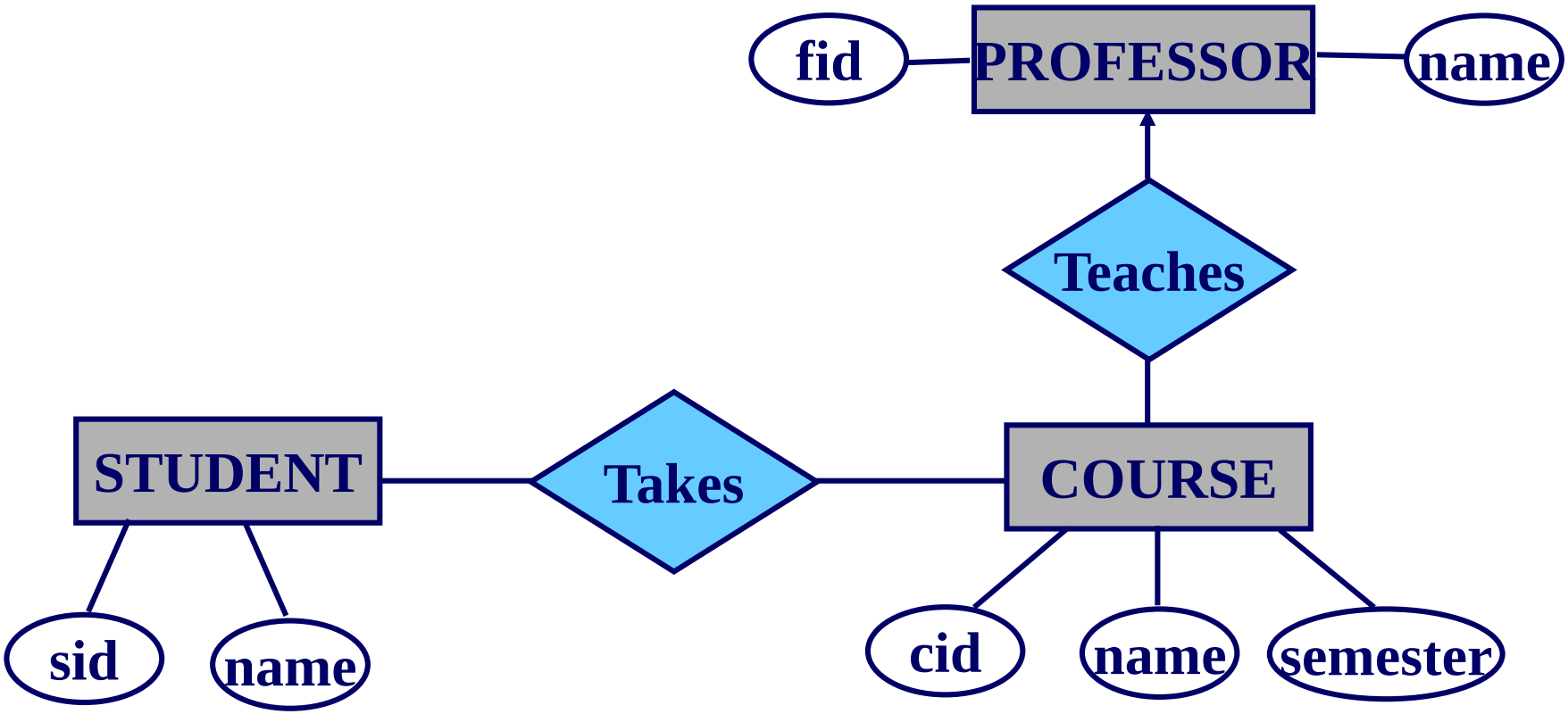
```
SELECT Name, Role, Rtype
FROM PERSON, ACTS_IN
WHERE MID = ‘1’ AND PERSON.PID == ACTS_IN.PID
```

Applications Use Queries in SQL

- Structured Query Language, often embedded (e.g., in Servlets, JSP, etc)
- Converted into a query plan that exploits properties; run over the data by the query optimizer and query execution engine

```
<html>
<body>
 <!– hypothetical Embedded SQL:
   SELECT Name, Role, Rtype
    FROM PERSON, ACTS_IN
    WHERE MID = ‘1’ AND
    PERSON.PID == ACTS_IN.PID
 -->
</body>
</html>
```

Another Example
Logical Model of a Database



Designing a Schema (Set of Relations)

STUDENT

sid	name
1	Jill
2	Bo
3	Maya

Takes

sid	cid
1	550-001
1	677-001
3	521-001

COURSE

cid	name	sem
550-001	DB	F11
677-001	Algo	F11
521-001	AI	S11

- Convert to tables + constraints
- Then need to do "physical" design: the layout on disk, indices, etc.

PROFESSOR

fid	name
1	Ives
2	Kannan
8	Ungar

Teaches

fid	cid
1	550-001
2	677-001
8	521-001

Database Systems

- The big commercial database vendors:
  - Oracle
  - IBM (with DB2)
  - Microsoft (SQL Server)
  - Sybase
- Some free database systems :
  - MySQL (acquired by Oracle..)
  - PostgreSQL

- We will use Oracle DB in this course

People who work with DBMSs

- Database Administrator DBA
  - Maintains databases, DBMS and related software
- Application Programmers
  - Software engineers (developers) that build software solutions for end users that access DBMS
- End Users
  - Example: bank teller uses “canned transactions”
- DBMS designers and implementers
  - Example: Oracle developers

New Trends in Databases

- Object-relational databases
- Main memory database systems
- XML
  - Relational databases with XML support
  - Native XML database systems
  - Lots of research on XML and databases
- Data integration
- Peer to peer, stream data management – still research
- BIG DATA
- NoSQL