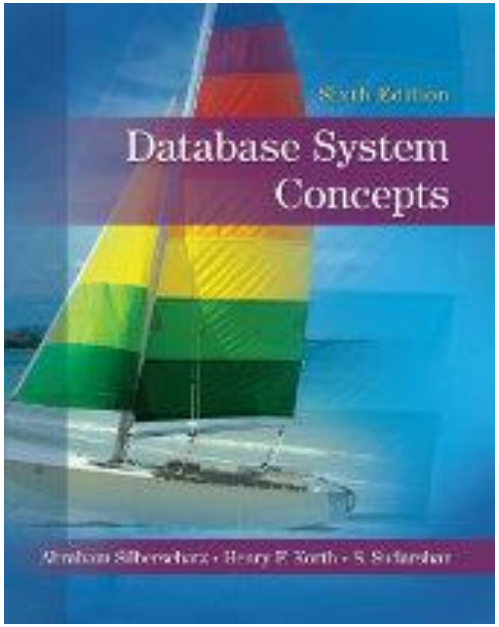
BİL 471 – Veritabanı Sistemleri

Introduction

Ders hakkında

- Web sayfası:
  http://web.cs.hacettepe.edu.tr/~ssen/teaching/bbm471.html
- Sunumları, bu sayfadan edinebilirsiniz.
- Bu dersten sonra hemen haber grubuna üye olacağım!

Textbook



- Textbook:
  - Database System Concepts, by Abraham Silberschatz, Henry F. Korth, and S. Sudarshan, McGraw Hill
- Türkçe Kaynaklar
  - Ünal Yarımağan "Veritabanı Sistemleri", Akademi Yayıncılık, 2. basım, 2010
- Other references:
  - Database Management Systems, by Raghu Ramakrishnan and Johannes Gehrke, McGraw-Hill
  - Database Systems: The Complete Book (2nd edition), by Hector Garcia-Molina, Jeffrey D. Ullman, and Jennifer D. Widom
  - Fundamentals of Database Systems, by Ramez Elmasri and Shamkant Navathe, Addison Wesley

SQL Readings

- Many SQL references available online
- Good online (free) SQL tutorials include:
  - A Gentle Introduction to SQL (http://sqlzoo.net/)
  - SQL for Web Nerds (http://philip.greenspun.com/sql/)

Tentative Grading

- Değerlendirme
  - Arasınav : %30 - Kapalı Kitap / Kapalı Notlar
  - Proje : %30
  - Genel Sınav : %40 - Kapalı Kitap / Kapalı Notlar

Course Overview

Introduction

E-R Models

Relational Model

Relational Algebra

SQL

Midterm exam

Relational Database Design

Normalization

Transactions

Concurrency

Other Topics

- NoSQL DB
- Object-oriented DB

Lecture 1: Introduction

What is a Database?

- A **database** is a large, integrated collection of data, typically describing the activities of one or more related organizations.

- For example, a university database might contain information about the following:
  - *Entities (Varlıklar)*
    - students, faculty, courses, and classrooms.
  - *Relationships* between entities *(Bağıntılar)*
    - students' enrollment in courses, faculty teaching courses, and the use of rooms for courses.

A Sample Relational Database

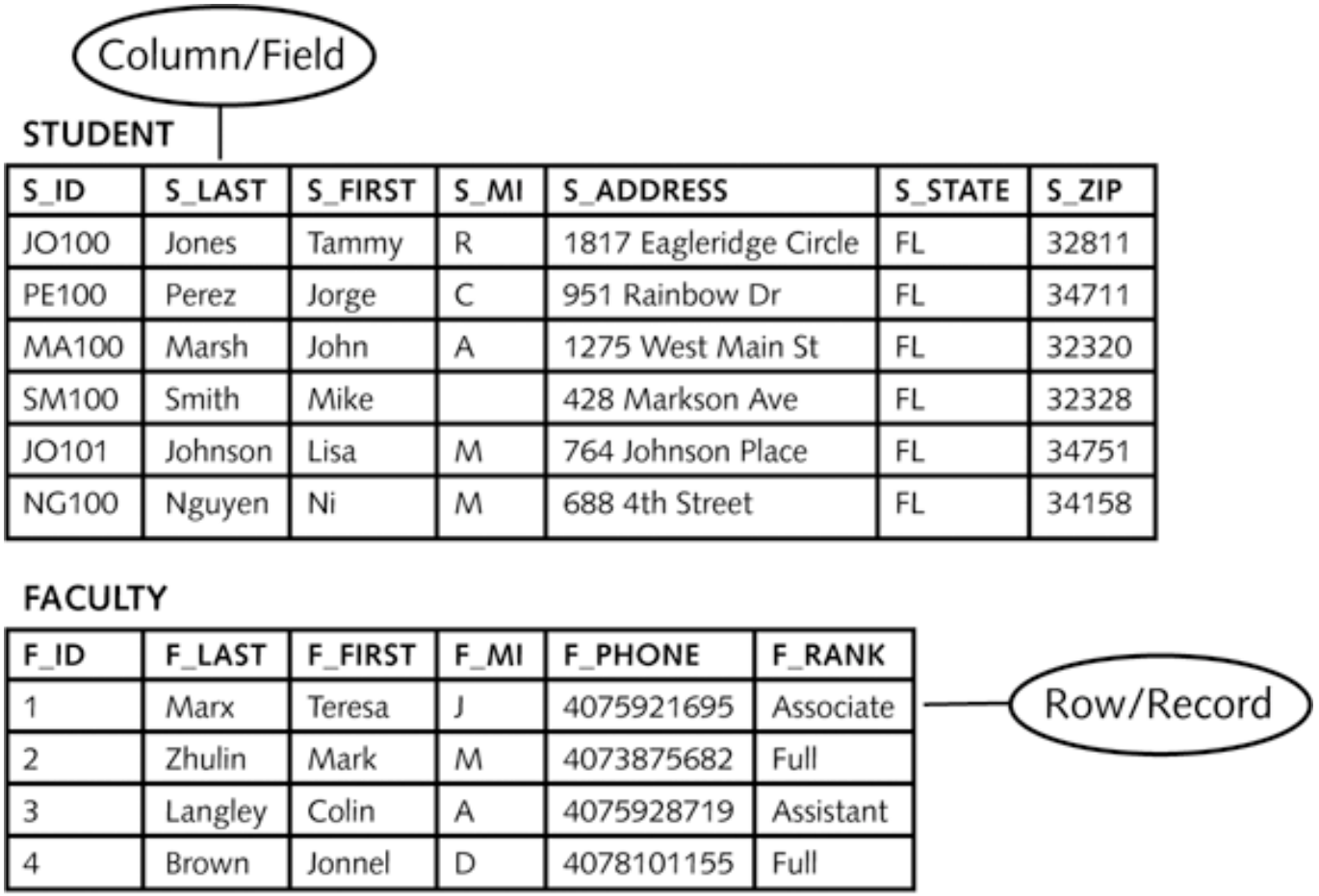


Figure 1-5 Examples of relational database tables

Your wallet is full of DB *records*…

- Identity Card
- Driver's license
- Credit cards
- University Card
- Medical insurance card
- Social security card
- Money (serial numbers)
- Photos (ids on back)
- Etc…

"You may not be interested in databases, but databases are interested in you." - Trotsky

Slide from M.P. Johnson NYU

What is a database management system (DBMS)?

Definition 1

A database management system, or DBMS, is software designed to assist in maintaining and utilizing large collections of data

Definition 2:

System for providing efficient, convenient, and safe multi-user storage of and access to massive amounts of persistent data

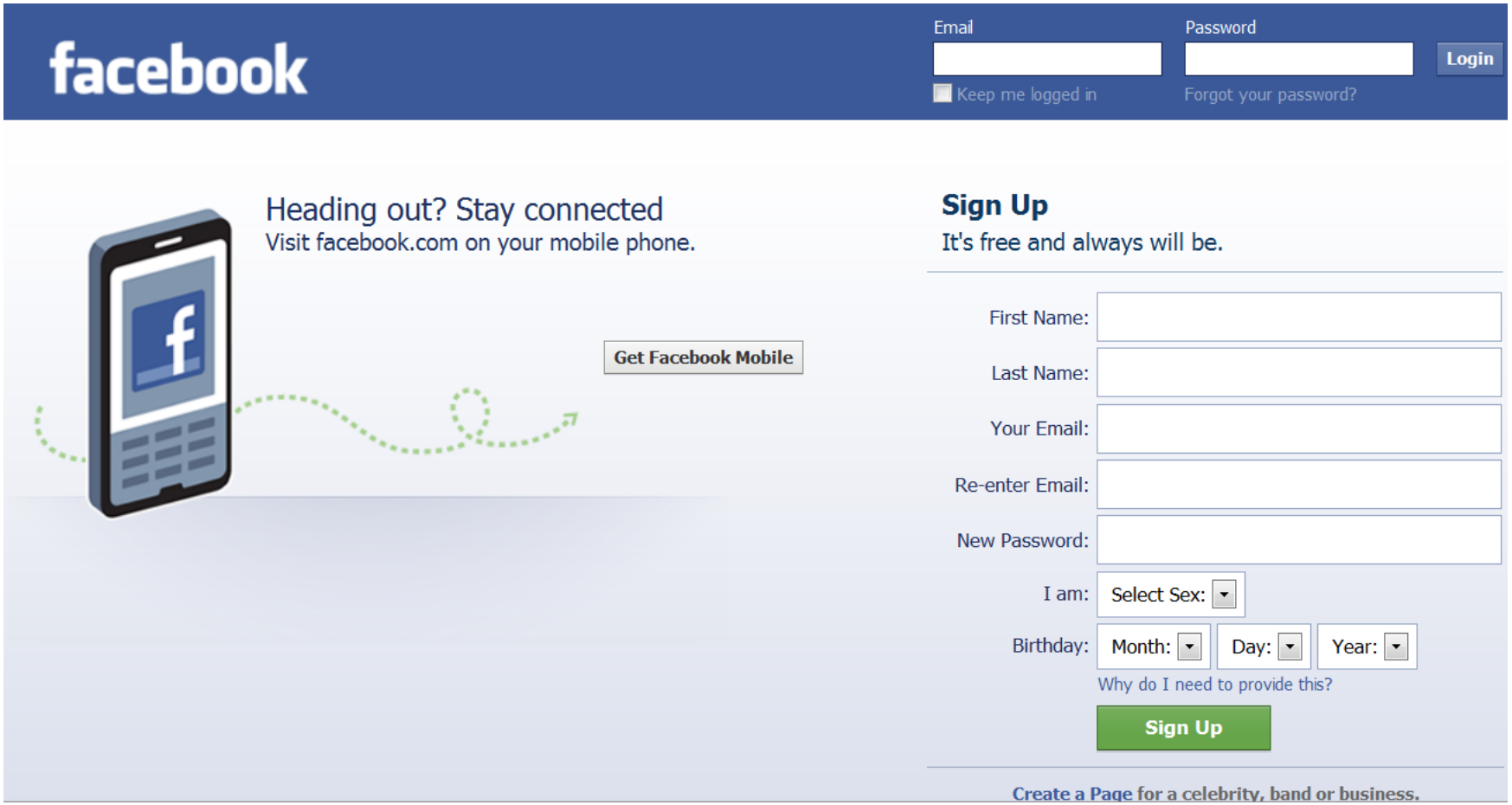
DBMS Examples

- Most familiar use: many Web sites rely heavily on DBMS's

- And many non-Web examples

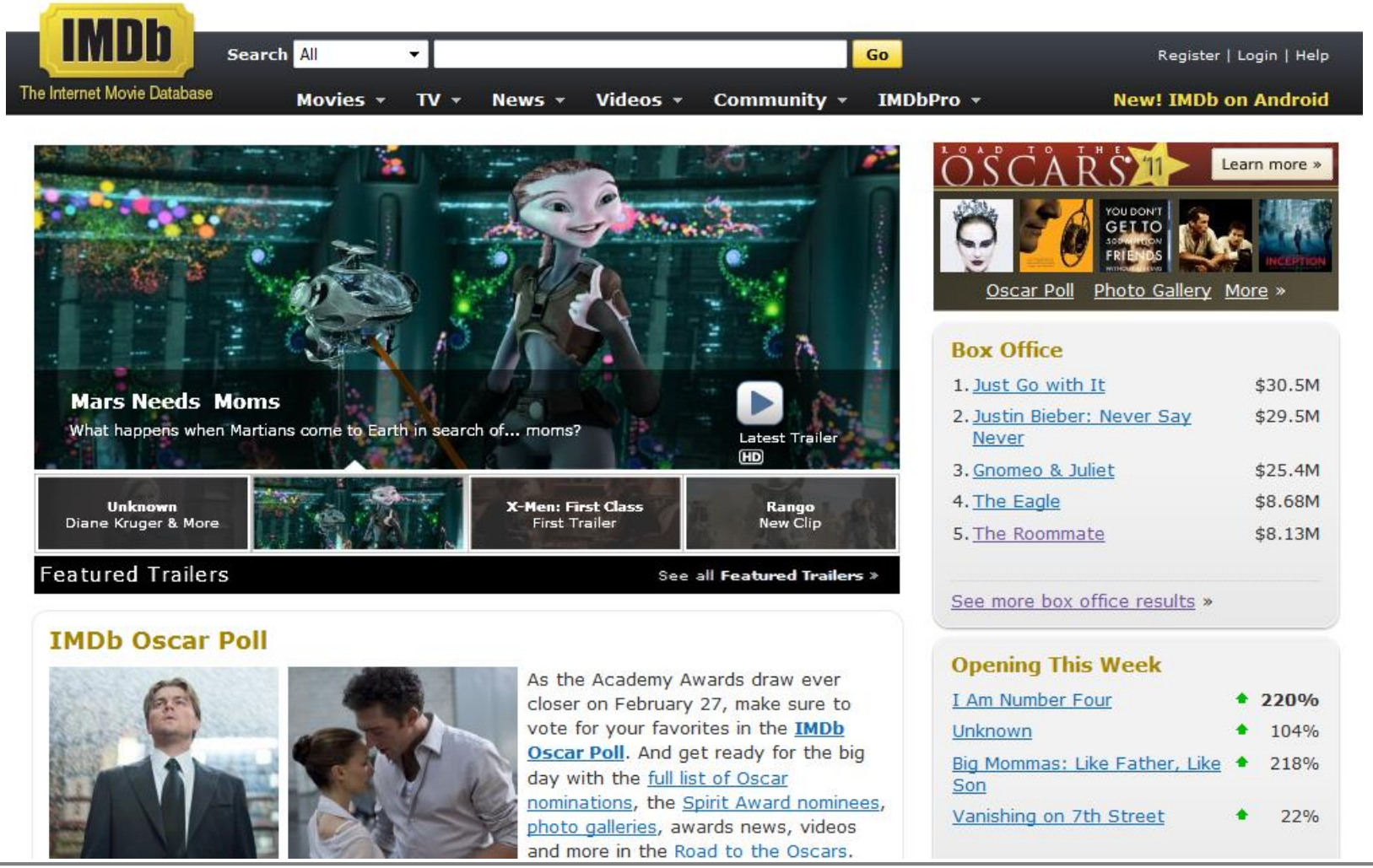
Frequently Used Sites With DBMS

Frequently Used Sites With DBMS



Frequently Used Sites With DBMS

Frequently Used Sites With DBMS



What is a database management system (DBMS)?

Definition 2:

System for providing efficient, convenient, and safe multi-user storage of and access to massive amounts of persistent data

Red words = key characteristics

Example: Banking system

- Data = information on accounts, customers, balances, current interest rates, transaction histories, etc.

- Massive: many gigabytes at a minimum for big banks, more if keep history of all transactions, even more if keep images of checks -> Far too big for memory

- Persistent: data outlives programs that operate on it

Slide from K.Minami

Two Perspectives in DB Systems

- User perspective: externals
  - how to use a database system?
  - conceptual data modeling, relational and other data models, database schema design, relational algebra, and the SQL query language.
- System perspective: internals
  - how to design and implement a database system?
  - data representation, indexing, query optimization and processing, transaction processing, concurrency control, and crash recovery

Slide from K.Minami

Example of a Traditional DB App

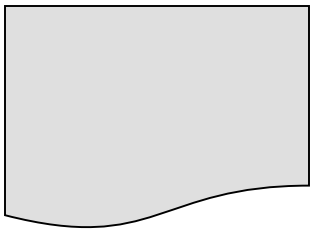
- Suppose we build a system

- We store:
  - checking accounts
  - savings accounts
  - account holders
  - state of each person's accounts

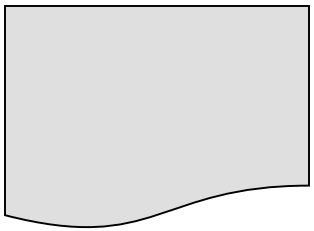
Can we do without a DBMS?

Sure! Start by storing the data in files:

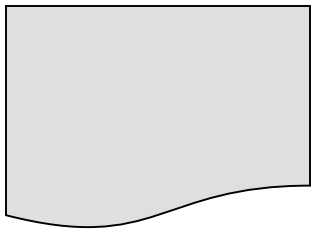
checking.txt



savings.txt



customers.txt



Now write C or Java programs to implement specific tasks…

Doing it without a DBMS...

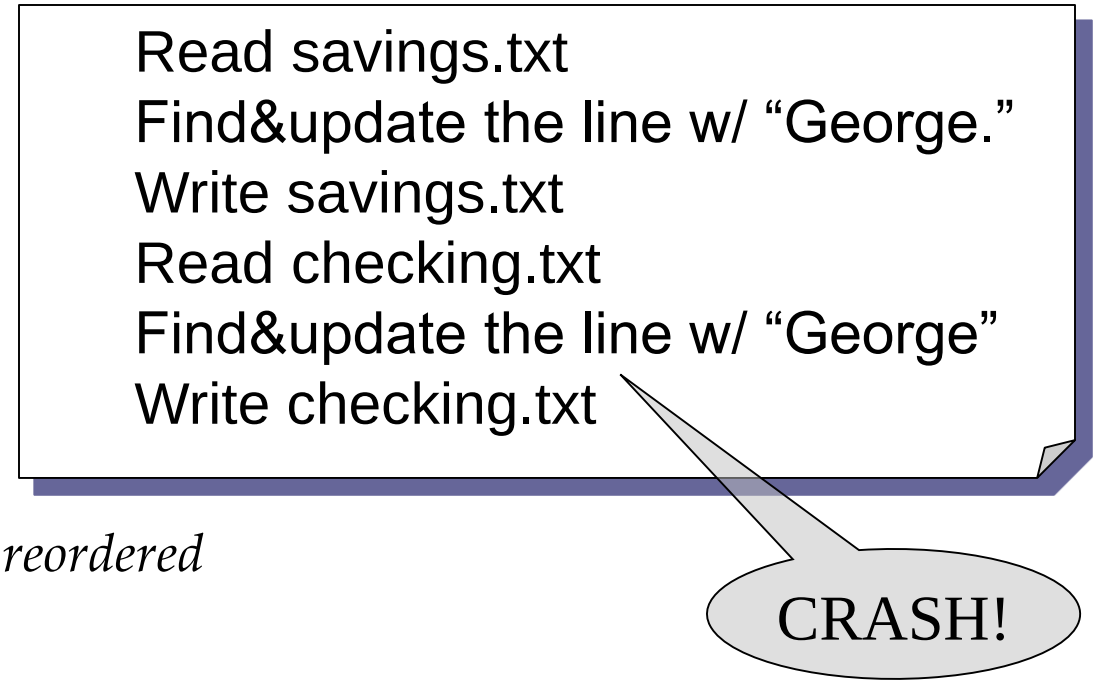
- Transfer $100 from George's savings to checking:

Write a C program to do the following:

```
Read savings.txt
Find&update the line w/"George"
    balance -= 100
Write savings.txt
Read checking.txt
Find&update the line w/"George"
    balance += 100
Write checking.txt
```

Problems without an DBMS...

1. System crashes:



   - Same problem *even if reordered*

2. Simultaneous access by many users
   - George and Dick visit ATMs at same time

Why is multi-user access hard?

Multi-user: many people/programs accessing same db, or even same data, simultaneously -> need careful controls

Alice @ ATM1: withdraw $100 from account #002

```
get balance from database;
if balance >= 100 then balance := balance - 100;
dispense cash;
put new balance into database;
```

Bob @ ATM2: withdraw $50 from account #002

```
get balance from database;
if balance >= 50 then balance := balance - 50;
dispense cash;
put new balance into database;
```

Initial balance = 200. Final balance = ??

Slide from K.Minami

Problems without a DBMS...

3. Large data sets (100s of GBs, or TBs, …)

- No indices
  - Finding "George" in huge flatfile is expensive

4. Modifications intractable without better data structures
  - "George" → "Georgie" is very expensive
  - Deletions are very expensive

Problems without an DBMS...

5. Security?
   - File system may lack security features

6. Application programming interface (API)?
   - Interfaces, interoperability

7. How to query the data?
   - need to write a new C++/Java program for every new query
   - need to worry about performance

Using File systems 1/2

Drawbacks of using file systems to store data:

- Data redundancy and inconsistency
  Multiple file formats, duplication of information in different files
- Difficulty in accessing data
  Need to write a new program to carry out each new task
- Data isolation — multiple files and formats
- Integrity problems
  Integrity constraints (e.g. account balance > 0) become part of program code
  Hard to add new constraints or change existing ones

Using File systems 2/2

- Atomicity of updates
  Failures may leave database in an inconsistent state with partial updates carried out
    E.g. transfer of funds from one account to another should either complete or not happen at all
- Concurrent access by multiple users
  Concurrent accessed needed for performance
  Uncontrolled concurrent accesses can lead to inconsistencies
    E.g. two people reading a balance and updating it at the same time
- Security problems

DBMSs were invented to solve all these problems!

Why use a DBMS?

- Data independence and efficient access
- Reduced application and development time
- Data integrity and security
- Uniform data administration
- Concurrent access, recovery from crashes

Back to the red words

- Safe:
  - from system failures
  - from malicious users
- Convenient:
  - simple commands to debit account, get balance, write statement, transfer funds, etc.
  - also unexpected queries should be easy
- Efficient:
  - don't scan the entire file to get balance of one account, get all accounts with low balances, get large transactions, etc.
  - massive data! -> DBMS's carefully tuned for performance

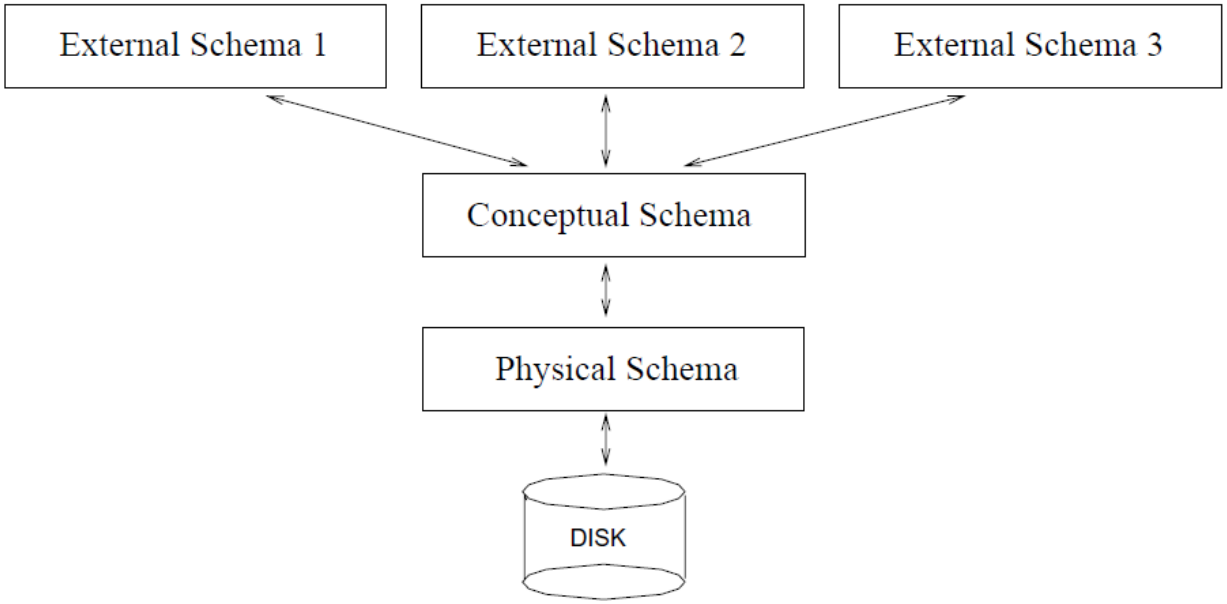
Slide from K.Minami

Schemas and Data

- Similar to types and variables in programming languages
- **Schema** – the logical structure of the database
  - schema: describes how data is to be structured, defined at set-up time, rarely changes
  - **Physical schema**: database design at the physical level
  - **Logical schema**: database design at the logical level
- **Instance** – the actual content of the database at a particular point in time
  - data is actual "instance" of database, changes rapidly
- **Physical Data Independence** – the ability to modify the physical schema without changing the logical schema
  - Applications depend on the logical schema
  - In general, the interfaces between the various levels and components should be well defined so that changes in some parts do not seriously influence others

Level of Abstractions in a DBMS

- Many views, single conceptual and physical schema.



Schemas

- Physical Schema
  - Describe the files and indexes used
- Conceptual Schema
  - Define logical structure
- External Schema (views)
  - Describes how user sees the data

Schema e.g.: college registrar

- Schema:
  - *Students*(*ssn: string, name: string, login: string, age: int, gpa: real)*
  - *Courses*(*cid: string, cname: string, credits: int)*
  - *Enrolled*(*ssn:string, cid:string, grade: string)*

- Physical schema:
  - Relations stored as unordered text files.
  - Indices on first column of each rel

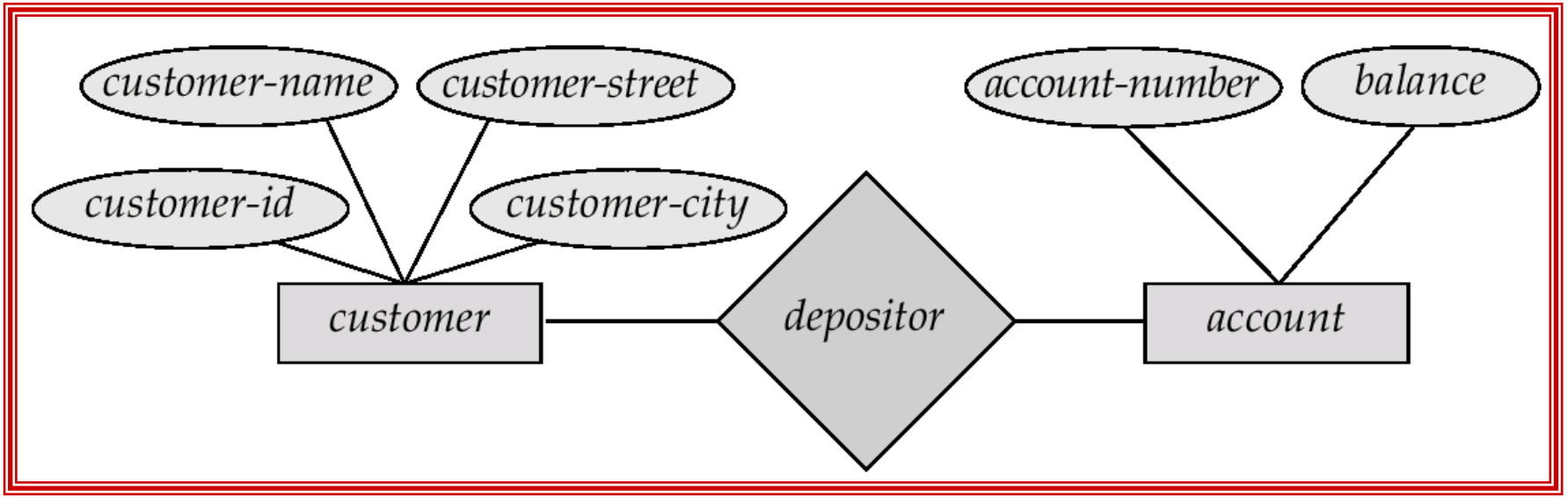
- Views:
  - *My_courses*(*cname: string, grade: string, credits: int)*
  - *Course_info*(*ssn: string, name: string, status: string)*

Slide from M.P. Johnson NYU

Data Models

- A collection of tools for describing
  - Data
  - Data relationships
  - Data semantics
  - Data constraints
- Relational model
- Entity-Relationship data model (mainly for database design)
- Object-based data models (Object-oriented and Object-relational)
- Semistructured data model (XML)
- Other older models:
  - Network model
  - Hierarchical model

Entity-Relationship Model



A sample E-R diagram

Relational Model

customer-id	*customer-name*	*customer-street*	*customer-city*
192-83-7465	Johnson	12 Alma St.	Palo Alto
019-28-3746	Smith	4 North St.	Rye
677-89-9011	Hayes	3 Main St.	Harrison
182-73-6091	Turner	123 Putnam Ave.	Stamford
321-12-3123	Jones	100 Main St.	Harrison
336-66-9999	Lindsay	175 Park Ave.	Pittsfield
019-28-3746	Smith	72 North St.	Rye

(a) The *customer* table

account-number	*balance*
A-101	500
A-215	700
A-102	400
A-305	350
A-201	900
A-217	750
A-222	700

(b) The *account* table

customer-id	*account-number*
192-83-7465	A-101
192-83-7465	A-201
019-28-3746	A-215
677-89-9011	A-102
182-73-6091	A-305
321-12-3123	A-217
336-66-9999	A-222
019-28-3746	A-201

(c) The *depositor* table

Database Queries

1. What is the name of the student with student id 123456?

2. What is the average salary of professors who teach the course with cid CS564?

3. How many students are enrolled in course CS564?

4. What fraction of students in course CS564 received a grade better than B?

5. Is any student with a GPA less than 3.0 enrolled in course CS564?

DDL and DML

- Data definition language (DDL)
  - commands for setting up schema of database

- Data Manipulation Language (DML)
  - Commands to manipulate data in database:
  - SELECT, INSERT, DELETE, MODIFY

  Also called "query language"

How the programmer sees the DBMS

- Start with SQL DDL to *create tables*:

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

- Continue with SQL to *populate tables:*

```
INSERT INTO Students
VALUES('Hillary', '123456789', 'undergraduate');
```

Slide from M.P. Johnson NYU

How the programmer sees the DBMS

Students:

SSN	Name	Category
123-45-6789	Hillary	undergrad
234-56-7890	Barak	grad
	…	…

Courses:

CID	**CName**
C20.0046	**Databases**
C20.0056	**Advanced Software**

Takes:

SSN	CID	semester
123-45-6789	C20.0046	Spring, 2004
123-45-6789	C20.0056	Spring, 2004
234-56-7890	C20.0046	Fall, 2003
	…	

Querying: Structured Query Language

- Find all the students who have taken **Bil354**

```
SELECT SSN FROM Takes
WHERE CID='Bil354';
```

- Find all the students who took **Bil354** *previously*:

```
SELECT SSN FROM Takes
WHERE CID=' Bil354 ' AND
Semester='Fall, 2009';
```

- Find the students' *names*:

```
SELECT Name FROM  Students, Takes
WHERE Students.SSN=Takes.SSN AND
CID=' Bil354 ' AND Semester='Fall, 2009';
```

Slide from M.P. Johnson NYU

People

- DBMS administrator (DBA)
  - Design of the conceptual and physical schemas
  - Security and authorization
  - Data availability and recovery from failures
  - Database tuning
- DBMS application programmers
- DBMS end-user: queries/modifies data

Slide from M.P. Johnson NYU

Database Industry

- Commercial DBMSs: Oracle, IBM's DB2, Microsoft's SQL Server, etc.
- Opensource: MySQL, PostgreSQL, etc.

- DBAs manage these
- Programmers write apps

- XML ("semi-structured data") also important

First Part of the Course: DBMS externals

- Entity-Relationship Model
- Relational Model
- Relational Database Design
- Relational Algebra
- SQL and DBMS Functionality:
  - SQL Programming
  - Queries and Updates
  - Indexes and Views
  - Constraints and Triggers

Second Part of the Course: DBMS internals

- Query Execution and Optimization
- Transaction Management
- Concurrency