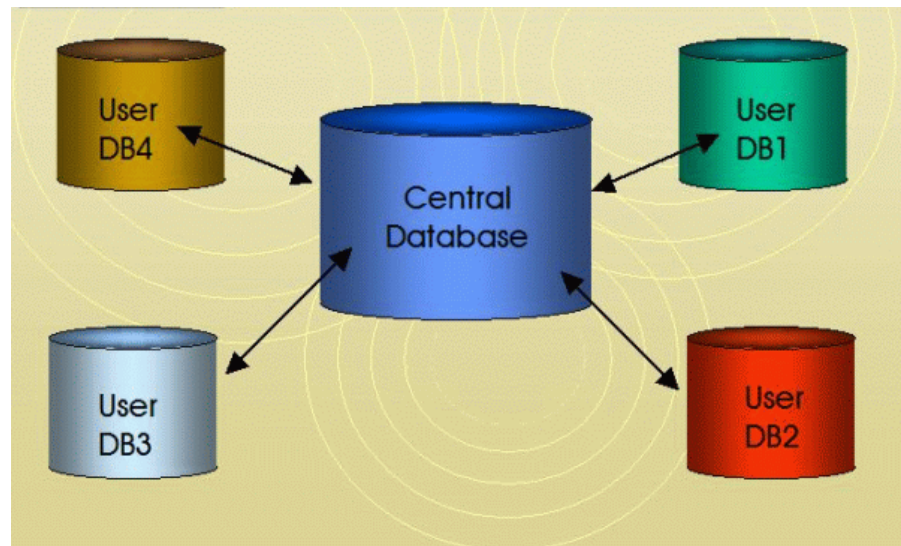


מידע במערכות תוכנה - חלק I

מסדי נתונים (Databases)



פיתוח מערכות תוכנה מבוססות Java
בית הספר למדעי המחשב, אוניברסיטת תל אביב

היום בשיעור

- נדון במידע ובייצוג במערכות תוכנה מורכבות
- החלק הראשון יעסוק בבסיסי נתונים (מסדי נתונים?)
- החלק השני (בשבוע הבא) יעסוק בטכנולוגיית XML

- חלק מהשקפים לקוחים מתוך הקורס במסדי נתונים של פרופסור טובה מילוא:
<http://www.cs.tau.ac.il/~milo> ■

What *Is* a Relational Database Management System ?

Database Management System = DBMS

Relational DBMS = RDBMS

- A collection of files that store the data
- A big C program written by someone else that accesses and updates those files for you

Where are RDBMS used ?

- ❑ Backend for traditional “database” applications:
 - CRM
 - ERP
 - Banking, insurance,...
- ❑ Backend for large Websites
- ❑ Backend for Web services

Example of a Traditional Database Application

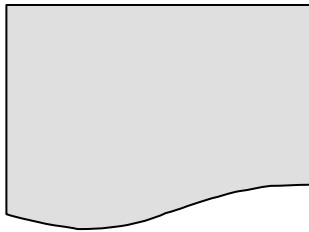
Suppose we are building a system to store the information about:

- students
- courses
- professors
- who takes what, who teaches what

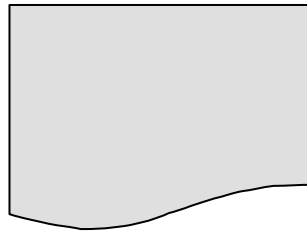
Can we do it without a DBMS ?

Sure we can! Start by storing the data in files:

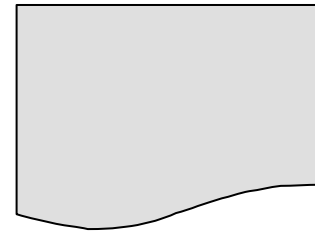
students.txt



courses.txt



professors.txt



Now write C or Java programs to implement specific tasks

Doing it without a DBMS...

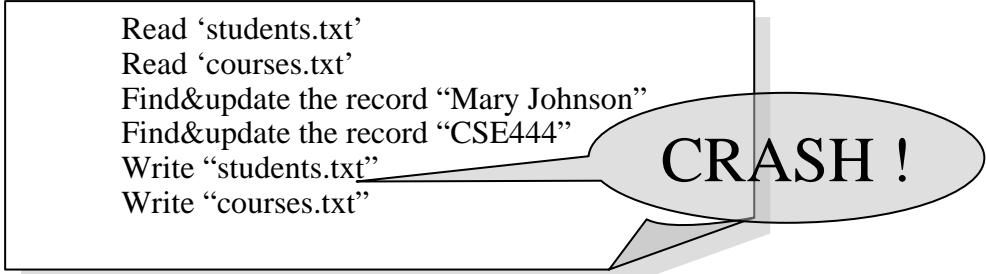
- Enroll “Mary Johnson” in “CSE444”:

Write a C program to do the following:

Read ‘students.txt’
Read ‘courses.txt’
Find&update the record “Mary Johnson”
Find&update the record “CSE444”
Write “students.txt”
Write “courses.txt”

Problems without an DBMS...

□ System crashes:



```
Read 'students.txt'  
Read 'courses.txt'  
Find&update the record "Mary Johnson"  
Find&update the record "CSE444"  
Write "students.txt"  
Write "courses.txt"
```

CRASH !

- What is the problem ?

□ Large data sets (say 50GB)

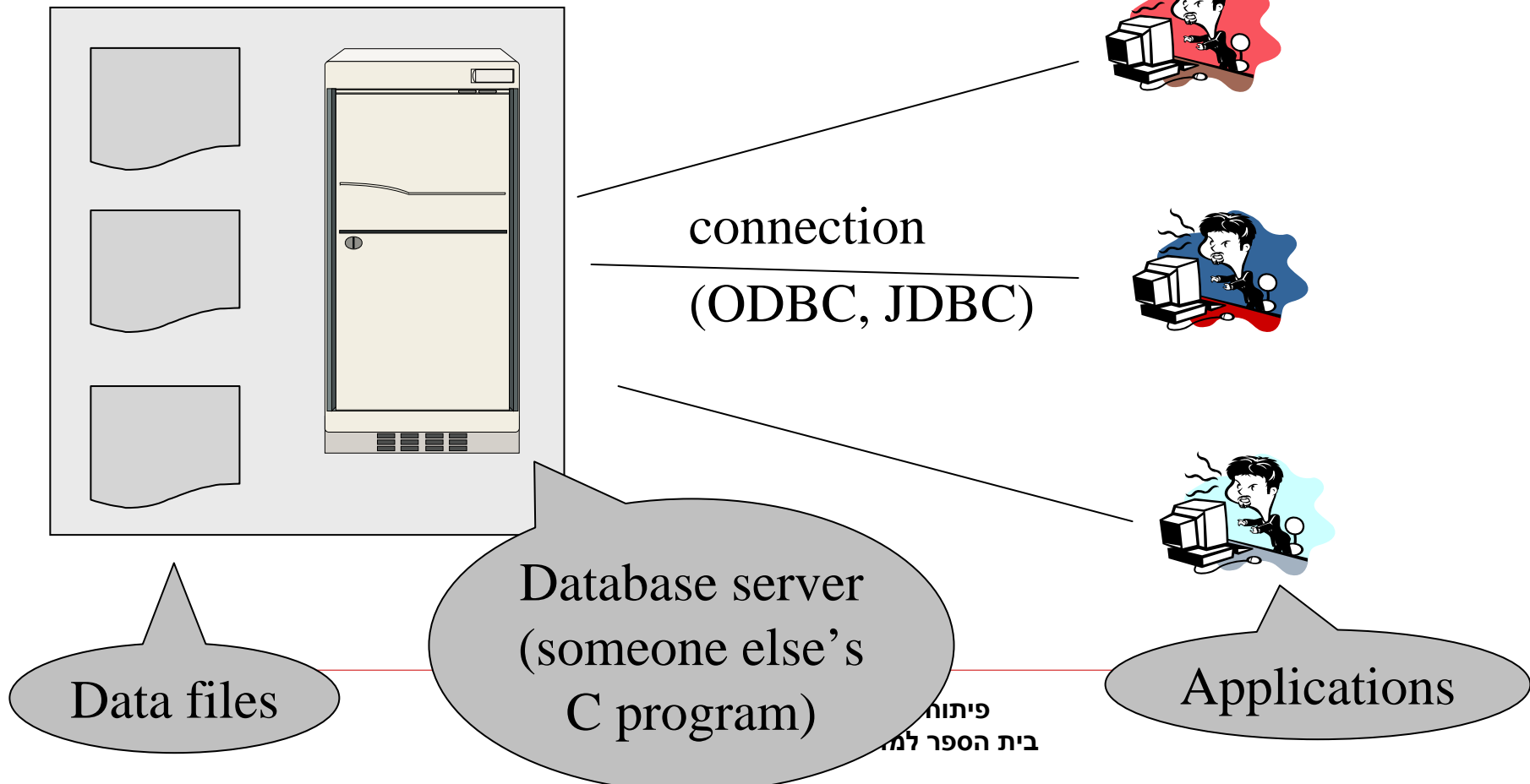
- What is the problem ?

□ Simultaneous access by many users

- Need locks: we know them from OS, but now data on disk; and is there any fun to re-implement them ?

Enters a DMBS

“Two tier database system”



Functionality of a DBMS

The programmer sees SQL, which has two components:

- Data Definition Language - DDL
- Data Manipulation Language - DML
 - query language

Behind the scenes the DBMS has:

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

Functionality of a DBMS

Two things to remember:

- Client-server architecture
 - Slow, cumbersome connection
 - But good for the data

- It is just someone else's C program
 - In the beginning we may be impressed by its speed
 - But later we discover that it can be frustratingly slow
 - We can do any particular task faster outside the DBMS
 - But the DBMS is *general* and *convenient*

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	CSE444
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 “Mary Johnson” in “CSE444”:

```
BEGIN TRANSACTION;

INSERT INTO Takes
  SELECT Students.SSN, Courses.CID
  FROM Students, Courses
  WHERE Students.name = 'Mary Johnson' and
         Courses.name = 'CSE444'

-- 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
 - C = consistency
 - I = independence
 - D = durability

Queries

- Find all courses that “Mary” takes

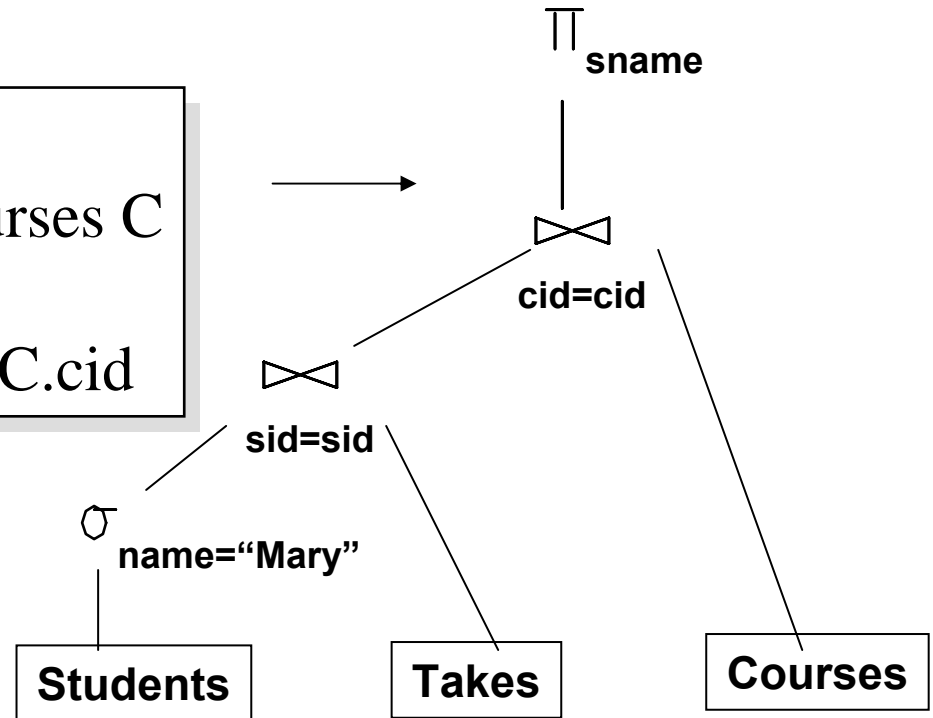
```
SELECT C.name
FROM   Students S, Takes T, Courses C
WHERE  S.name=“Mary” and
       S.ssn = T.ssn and T.cid = C.cid
```

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

Queries, behind the scene

Declarative SQL query \longrightarrow *Imperative query execution plan:*

```
SELECT C.name
FROM Students S, Takes T, Courses C
WHERE S.name="Mary" and
      S.ssn = T.ssn and T.cid = C.cid
```



The **optimizer** chooses the best execution plan for a query

Database Systems

- The big commercial database vendors:
 - Oracle
 - IBM (with DB2) bought Informix recently
 - Microsoft (SQL Server)
 - Sybase

- Some free database systems:
 - Postgres
 - MySQL
 - Predator

- Here we use MySQL. You may use something else, but then you are on your own.

New Trends in Databases

- ❑ Object-relational databases
- ❑ Main memory database systems
- ❑ XML XML XML !
 - Relational databases with XML support
 - Middleware between XML and relational databases
 - Native XML database systems
 - Lots of research here at TAU on XML and databases
- ❑ Data integration
- ❑ Peer to peer, stream data management – still research

Textbook(s)

Main textbook, to be available at the bookstore:

- ❑ *Database Systems: The Complete Book*, Hector Garcia-Molina, Jeffrey Ullman, Jennifer Widom

Almost identical:

- ❑ *A First Course in Database Systems*, Jeff Ullman and Jennifer Widom
- ❑ *Database Implementation*, Hector Garcia-Molina, Jeff Ullman and Jennifer Widom

Other Texts

- *Database Management Systems*, Ramakrishnan
 - very comprehensive

- *Fundamentals of Database Systems*, Elmasri, Navathe
 - very widely used

- *Foundations of Databases*, Abiteboul, Hull, Vianu
 - Mostly theory of databases

- *Data on the Web*, Abiteboul, Buneman, Suciu
 - XML and other new/advanced stuff

Other Readings

Reading from the Web:

- **SQL for Web Nerds**, by Philip Greenspun,
<http://philip.greenspun.com/sql/>