Carnegie Mellon Univ.  
Dept. of Computer Science  
15-415/615 - DB Applications  

C. Faloutsos  
Lecture#1: Introduction

Outline

• Introduction to DBMSs  
• The Entity Relationship model  
• The Relational Model  
• SQL: the commercial query language  
• DB design: FD, 3NF, BCNF  
• indexing, q-opt  
• concurrency control & recovery  
• advanced topics (data mining, multimedia)

We’ll learn:

• What are RDBMS  
  – when to use them  
  – how to model data with them  
  – how to store and retrieve information  
  – how to search quickly for information  
• Internals of an RDBMS: indexing, transactions
We’ll learn (cnt’d)

• Advanced topics
  – multimedia indexing (how to find similar, eg., images)
  – data mining (how to find patterns in data)

Administrivia

• Weights: as announced

  Course grade

  30% ASGN
  30% MT
  40% Final exam

  Sum = 100%

  ASGN1 ... ASGN8

Administrivia - II

• FYI: ASGN3 and ASGN7 are heavy
• Late policy: 4 ‘slip days’
• Exams: no aids allowed, except
  – 1 page with your notes (both sides) for MT
  – 2 such pages for Final
Detailed outline

- Introduction
  - Motivating example
  - How do DBMSs work? DDL, DML, views.
  - Fundamental concepts
  - DBMS users
  - Overall system architecture
  - Conclusions

What is the goal of rel. DBMSs

(eg., you have 50 friends + phone#;
Or a dentist has 100 customers, addresses, visit-info, treatment-info)
How can RDBMSs help?

What is the goal of rel. DBMSs

Electronic record-keeping:
**Fast and convenient** access to information.
Definitions

• ‘DBMS’ = ‘Data Base Management System’:
  the (commercial) system, like:
  DB2, Oracle, MS SQL-server, ...
• ‘Database system’: DBMS + data + application programs

Motivating example

Eg.: students, taking classes, obtaining grades;
• find my gpa
• <and other ad-hoc queries>

Obvious solution: paper-based

• advantages?
• disadvantages?

eg., student folders,
alpha sorted
Obvious solution: paper-based

- advantages?
  - cheap; easy to use
- disadvantages?
  - eg., student folders, alpha sorted

Obvious solution: paper-based

- advantages?
  - cheap; easy to use
- disadvantages?
  - no 'ad hoc' queries
  - no sharing
  - large physical foot-print

Next obvious solution

- computer-based (flat) files +
- C (Java, ...) programs to access them

  e.g., one (or more) UNIX/DOS files, with student records and their courses
Next obvious solution

your layout for the student records?

(eg., comma-separated values 'csv')
  Smith,John,123,db,A,os,B
  Tompson,Peter,234
  Atkinson,Mary,345,os,B,graphics,A
Problems?

- inconvenient access to data (need ‘C++’ expertize, plus knowledge of file-layout)
  - data isolation
- data redundancy (and inconcistencies)
- integrity problems
- atomicity problems

Problems? (cont’d)

- ... 
- concurrent-access anomalies 
- security problems
Problems? (cont’d)

[ why?
because of two main reasons:
  – file-layout description is buried within the C
    programs and
  – there is no support for transactions
    (concurrency and recovery)
]

DBMSs handle exactly these two problems

DBMS solution

• commercial/freeware DBMS &
• application programs

Main vendors/products

Commercial
• Oracle
• IBM/DB2
• MS SQL-server
• Sybase
• (MS Access,
• …)

Open source
Postgres (UCB)
mySQL
sqlite (sqlite.org)
miniBase (Wisc)
Predator (Cornell)

(www.acm.org/sigmod)
<Demo with sqlite3>

• Insert ‘student’ and ‘takes’ records
• Find the ‘os’ class roster
• Find the GPA of ‘Smith’

www.cs.cmu.edu/~christos/courses/dbms.S13/files/sqldemo.zip

Detailed outline

• Introduction
  – Motivating example
  – How do DBMSs work? DDL, DML, views.
  – Fundamental concepts
  – DBMS users
  – Overall system architecture
  – Conclusions

How do DBs work?

Pictorially:

DBMS

select * from student

and meta-data = catalog = data dictionary
How do DBs work?

% sqlite3 miniu.sql
sqlite>create table student ( 
    ssn fixed; 
    name char(20) );

Smith,John, 123,db,A,os,B
Tompson,Peter,234
Atkinson,Mary,345,os,B,graphics,A
create table student (ssn fixed, name char(20));
insert into student values(123, "Smith");
insert into student values(234, "Tompson");
insert into student values(345, "Atkinson");

-- see what we have inserted
select * from student;

ssn   name
------- -------
123   Smith
234   Tompson
345   Atkinson

How do DBs work?

sqlite>create table takes (ssn fixed, cid char(10), grade fixed);

-- register students in classes and give them grades
drop table if exists takes;
cREATE table takes (ssn fixed, cid char(10), grade fixed);
insert into takes values(123, "db", 4);
insert into takes values(123, "os", 3);
insert into takes values(345, "os", 3);
insert into takes values(345, "graphics", 4);

Smith,John,123,db,A
Tompson,Peter,234
Atkinson,Mary,345,os,B,graphics,A
-- see what we inserted

select * from takes;

<table>
<thead>
<tr>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>db</td>
<td>4</td>
</tr>
<tr>
<td>123</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>graphics</td>
<td>4</td>
</tr>
</tbody>
</table>

Smith, John, 123, db, A, os, B
Tompson, Peter, 234
Atkinson, Mary, 345, os, B, graphics, A

How do DBs work - cont’d

More than one tables - joins
Eg., roster (names only) for ‘os’

```
sqlite> select name from student, takes
    where student.ssn = takes.ssn
    and takes.c-id = 'os'
```

-- see what we inserted

select * from takes;

<table>
<thead>
<tr>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>db</td>
<td>4</td>
</tr>
<tr>
<td>123</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>graphics</td>
<td>4</td>
</tr>
</tbody>
</table>

Smith, John, 123, db, A, os, B
Tompson, Peter, 234
Atkinson, Mary, 345, os, B, graphics, A
-- find the os class roster

select name from student, takes
where student.ssn = takes.ssn
and cid="os";

name
---------
Smith
Atkinson

Views - a powerful tool!

what and why?

• suppose secy is allowed to see only ssn’s and GPAs, but not individual grades
• -> VIEWS!

Views

sqlite> create view fellowship as (select ssn, avg(grade)
from takes group by ssn);
### Views

Views = ‘virtual tables’

**sqlite**> select * from fellowship;

<table>
<thead>
<tr>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>db</td>
<td>4</td>
</tr>
<tr>
<td>123</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>graphics</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssn</th>
<th>avg(grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>3.5</td>
</tr>
<tr>
<td>345</td>
<td>3.5</td>
</tr>
</tbody>
</table>

**sql**> grant select on fellowship to secy;

('grant' not supported in sqlite)

<table>
<thead>
<tr>
<th>ssn</th>
<th>cid</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>db</td>
<td>4</td>
</tr>
<tr>
<td>123</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>os</td>
<td>3</td>
</tr>
<tr>
<td>345</td>
<td>graphics</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>ssn</th>
<th>avg(grade)</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>3.5</td>
</tr>
<tr>
<td>345</td>
<td>3.5</td>
</tr>
</tbody>
</table>
Iterating: advantages over (flat) files

- logical and physical data independence, because data layout, security etc info: stored explicitly on the disk
- concurrent access and transaction support

Disadvantages over (flat) files?

- Price
- additional expertise (SQL/DBA)
  hence: over-kill for small, single-user data sets

But: mobile phones (eg., android) use sqlite; some versions of firefox do, too: 
  /mozilla/.../cookies.sqlite etc
Detailed outline

• Introduction
  – Motivating example
  – How do DBMSs work? DDL, DML, views.
• Fundamental concepts
  – DBMS users
  – Overall system architecture
  – Conclusions

Fundamental concepts

• 3-level architecture
• logical data independence
• physical data independence

3-level architecture

• view level
• logical level
• physical level
3-level architecture

- view level
- logical level: eg., tables
  - STUDENT(ssn, name)
  - TAKES (ssn, cid, grade)
- physical level:
  - how are these tables stored, how many bytes / attribute etc

3-level architecture

- view level, eg:
  - v1: select ssn from student
  - v2: select ssn, c-id from takes
- logical level
- physical level

3-level architecture

- view level -> ‘fellowship’
- logical level -> ‘student’ ‘takes’
- physical level -> indices, hash, …
3-level architecture

• -> hence, **physical** and **logical** data independence:
• logical D.I.:
  – ???
• physical D.I.:
  – ???

Detailed outline

• Introduction
  – Motivating example
  – How do DBMSs work? DDL, DML, views.
  – Fundamental concepts
• DBMS users
  – Overall system architecture
  – Conclusions
Database users

- ‘naive’ users
- casual users
- application programmers
- [DBA (Data base administrator)]

Casual users

```
select *
from student
```

Naive users

Pictorially:

app. (eg., report generator)
App. programmers

• those who write the applications (like the 'report generator')

DB Administrator (DBA)

• Duties?

DBMS

app. (eg., report generator)

data

and meta-data = catalog

DB Administrator (DBA)

• Duties?

DBMS

app. (eg., report generator)

data

and meta-data = catalog
DB Administrator (DBA)

- schema definition ('logical' level)
- physical schema (storage structure, access methods)
- schema modifications
- granting authorizations
- integrity constraint specification

Detailed outline

- Introduction
  - Motivating example
  - How do DBMSs work? DDL, DML, views.
  - Fundamental concepts
  - DBMS users
  - Overall system architecture
  - Conclusions

Overall system architecture

- [Users]
- DBMS
  - query processor
  - storage manager
- [Files]
Overall system architecture

• query processor
  – DML compiler
  – embedded DML pre-compiler
  – DDL interpreter
  – Query evaluation engine

Overal system architecture (cont’d)

• storage manager
  – authorization and integrity manager
  – transaction manager
  – buffer manager
  – file manager
Overall system architecture (cont’d)

- Files
  - data files
  - data dictionary = catalog (= meta-data)
  - indices
  - statistical data

Some examples:

- DBA doing a DDL (data definition language) operation, eg.,
  create table student ...

[Diagram of system architecture showing various components like DML int., DML proc, query eval., app. pgm(o), DDL int., query proc., trans. mgr, buff. mgr, file mgr, meta-data, data, naive, app. pgmr, casual, DBA, users]
Some examples:

- casual user, asking for an update, eg.:
  update student
  set name to 'smith'
  where ssn = '345'
Some examples:

• app. programmer, creating a report, eg
  
  main()
  
  
  ...  
  
  exec sql “select * from student”
  
  ...
  
  }

• app. programmer, creating a report, eg
  
  main()
  
  
  ...  
  
  exec sql “select * from student”
  
  ...
  
  }
Some examples:

- ‘naive’ user, running the previous app.

Detailed outline

- Introduction
  - Motivating example
  - How do DBMSs work? DDL, DML, views.
  - Fundamental concepts
  - DBMS users
  - Overall system architecture
  - Conclusions
Conclusions

- (relational) DBMSs: electronic record keepers
- customize them with `create table` commands
- ask SQL queries to retrieve info

Conclusions cont’d

main advantages over (flat) files & scripts:
- **logical + physical data independence** (ie., flexibility of adding new attributes, new tables and indices)
- **concurrency control** and **recovery**