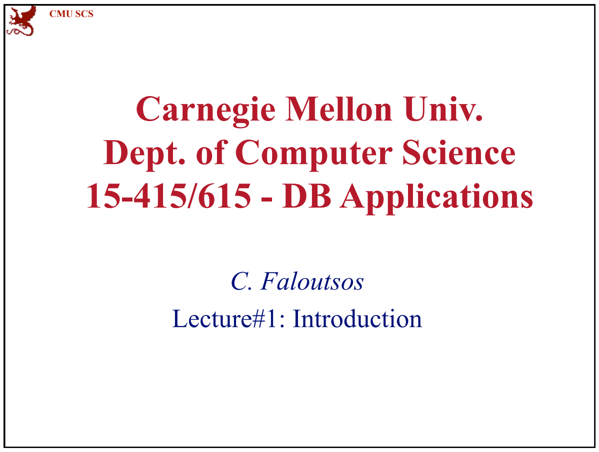
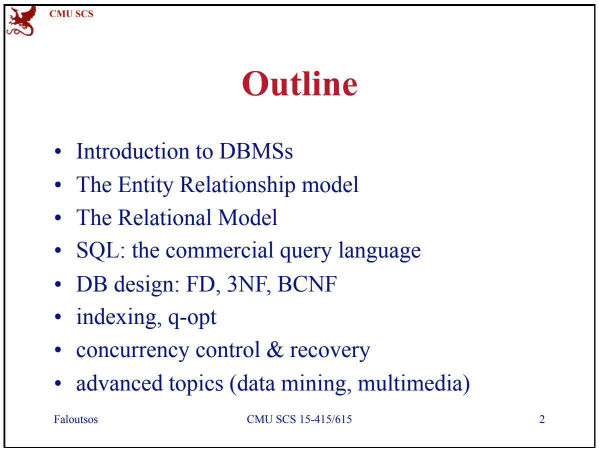


CMU SCS

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

C. Faloutsos

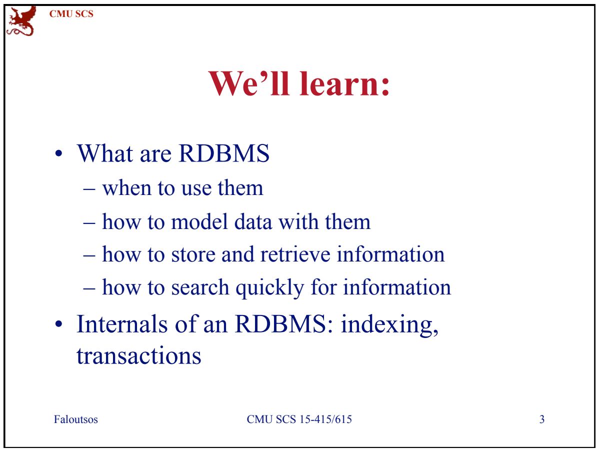
Lecture#1: Introduction



CMU SCS

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)



CMU SCS

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

CMU SCS

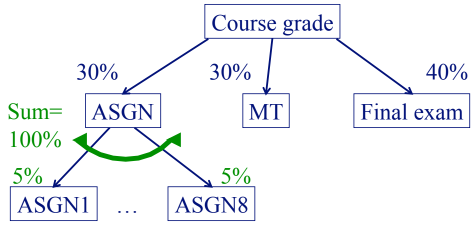
We'll learn (cnt'd)

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

CMU SCS

Administrivia

- Weights: as announced



Course grade

30% ASGN

30% MT

40% Final exam

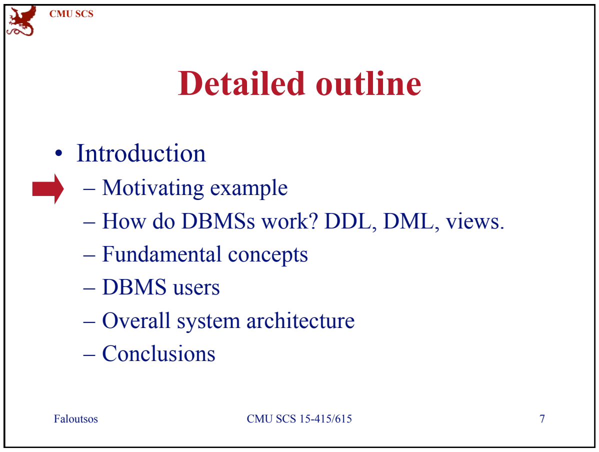
Sum= 100%

5% ASGN1 … 5% ASGN8

CMU SCS

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

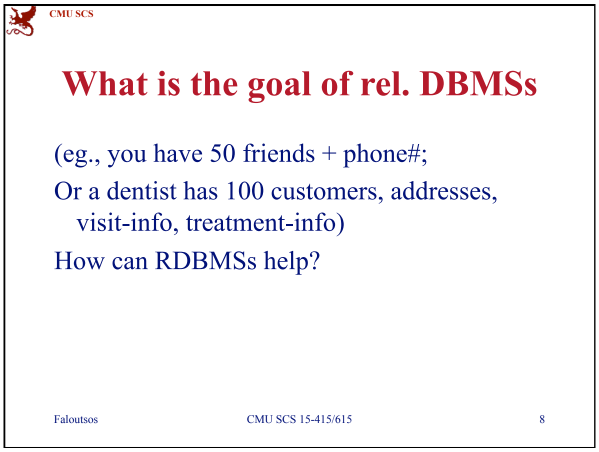


CMU SCS

Detailed outline

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

Faloutsos CMU SCS 15-415/615 7



CMU SCS

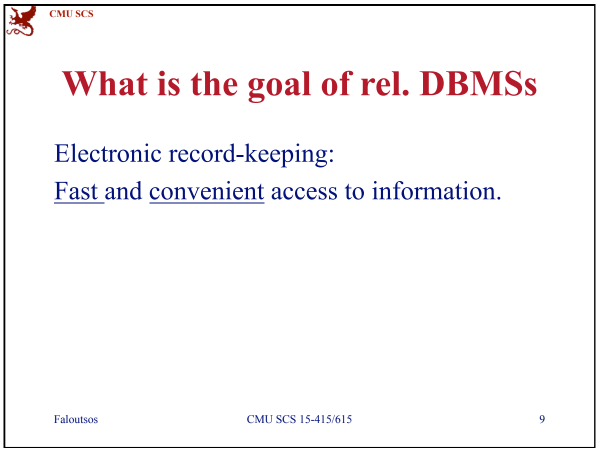
What is the goal of rel. DBMSs

(eg., you have 50 friends + phone#;

Or a dentist has 100 customers, addresses, visit-info, treatment-info)

How can RDBMSs help?

Faloutsos CMU SCS 15-415/615 8



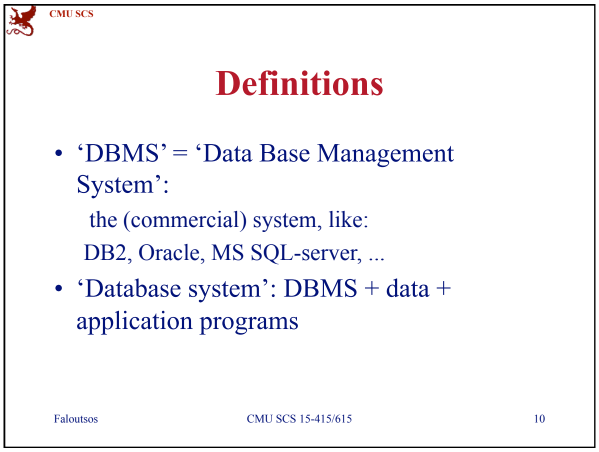
CMU SCS

What is the goal of rel. DBMSs

Electronic record-keeping:

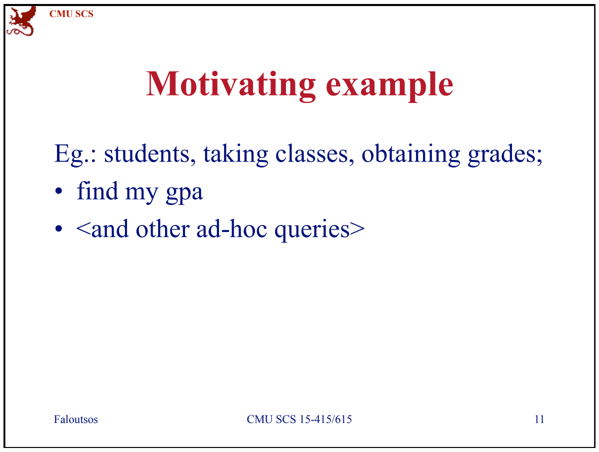
Fast and convenient access to information.

Faloutsos CMU SCS 15-415/615 9



Definitions

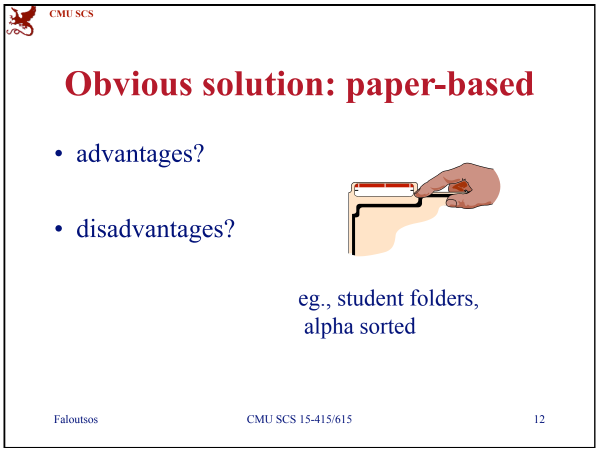
- ‘DBMS’ = ‘Data Base Management System’:
  the (commercial) system, like:
  DB2, Oracle, MS SQL-server, ...
- ‘Database system’: DBMS + data + application programs



Motivating example

Eg.: students, taking classes, obtaining grades;

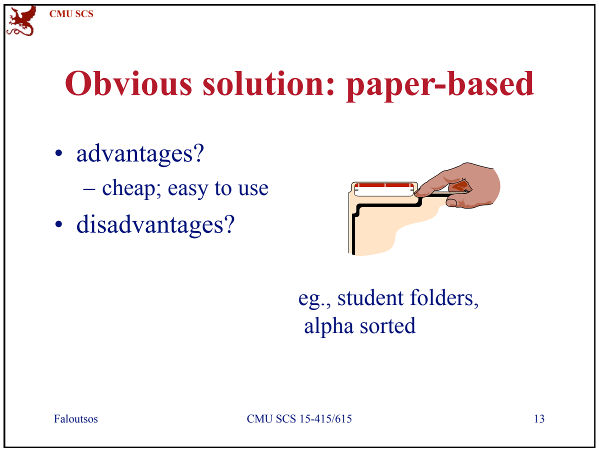
- find my gpa
- <and other ad-hoc queries>



Obvious solution: paper-based

- advantages?
- disadvantages?

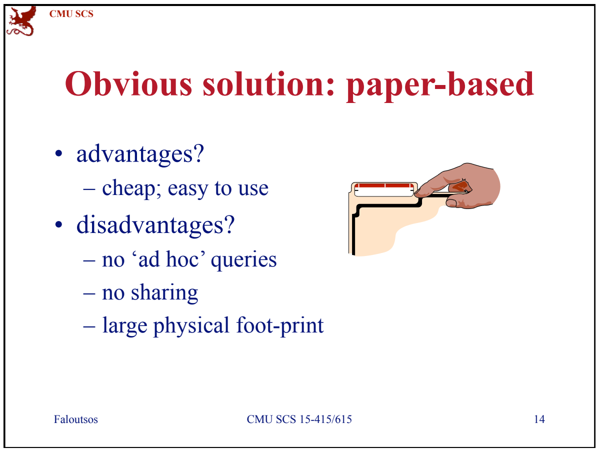
eg., student folders,
alpha sorted



Obvious solution: paper-based

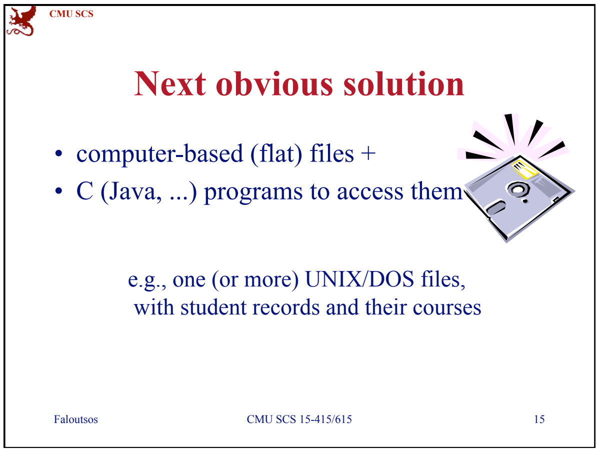
- advantages?
  - cheap; easy to use
- disadvantages?

eg., student folders,
alpha sorted



Obvious solution: paper-based

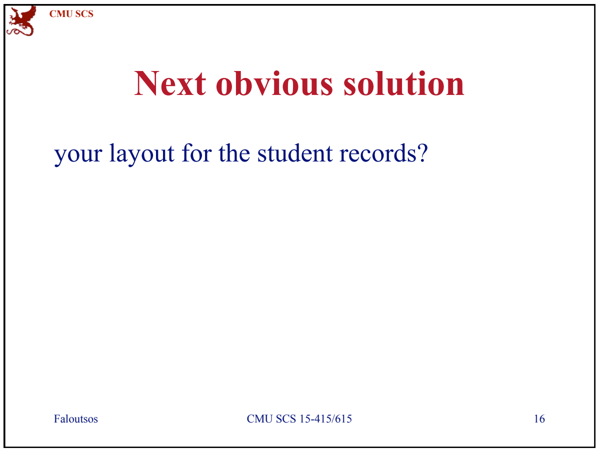
- advantages?
  - cheap; easy to use
- disadvantages?
  - no ‘ad hoc’ queries
  - no sharing
  - large physical foot-print



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

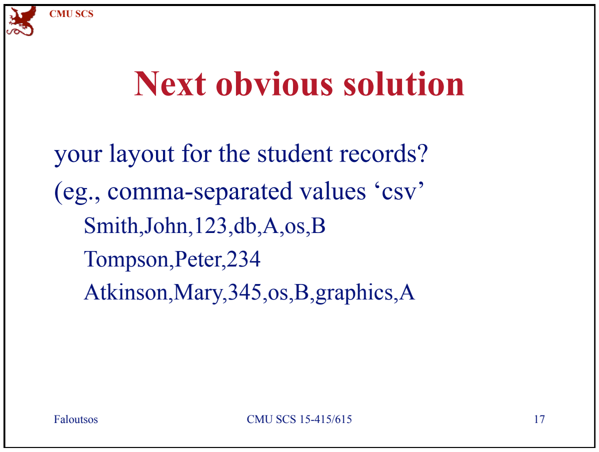


CMU SCS

Next obvious solution

your layout for the student records?

Faloutsos CMU SCS 15-415/615 16



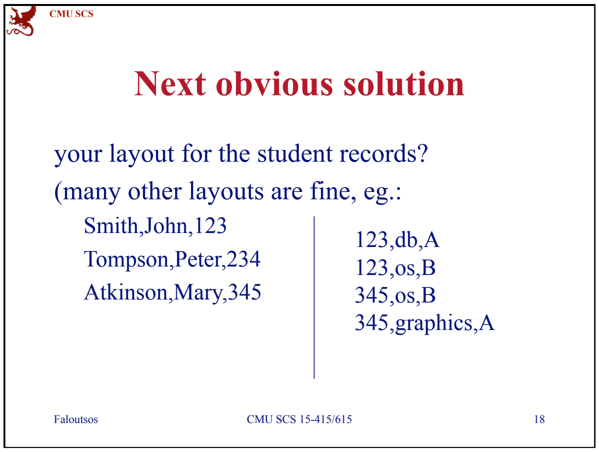
CMU SCS

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

Faloutsos CMU SCS 15-415/615 17



CMU SCS

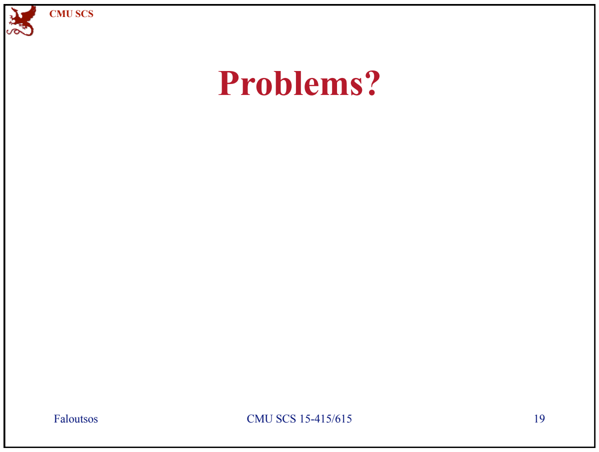
Next obvious solution

your layout for the student records?
(many other layouts are fine, eg.:

Smith,John,123
Tompson,Peter,234
Atkinson,Mary,345

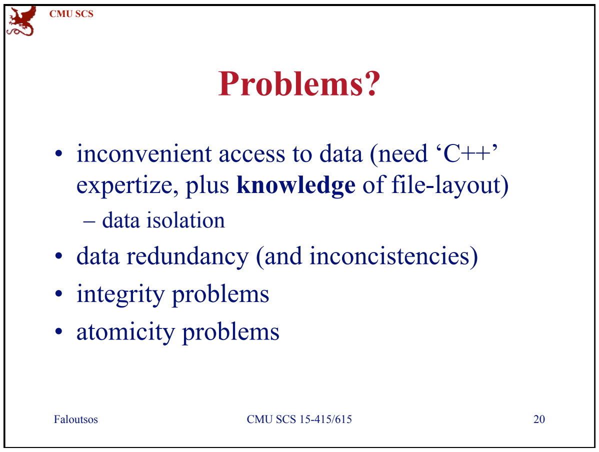
123,db,A
123,os,B
345,os,B
345,graphics,A

Faloutsos CMU SCS 15-415/615 18



CMU SCS

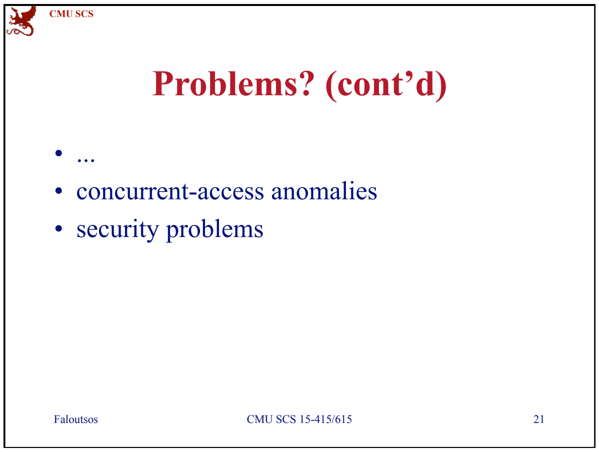
Problems?



CMU SCS

Problems?

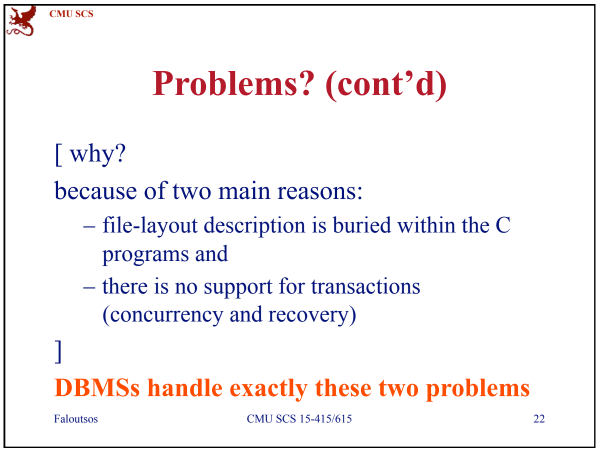
- inconvenient access to data (need ‘C++’ expertize, plus **knowledge** of file-layout)
  - data isolation
- data redundancy (and inconcistencies)
- integrity problems
- atomicity problems



CMU SCS

Problems? (cont’d)

- ...
- concurrent-access anomalies
- security problems



CMU SCS

Problems? (cont'd)

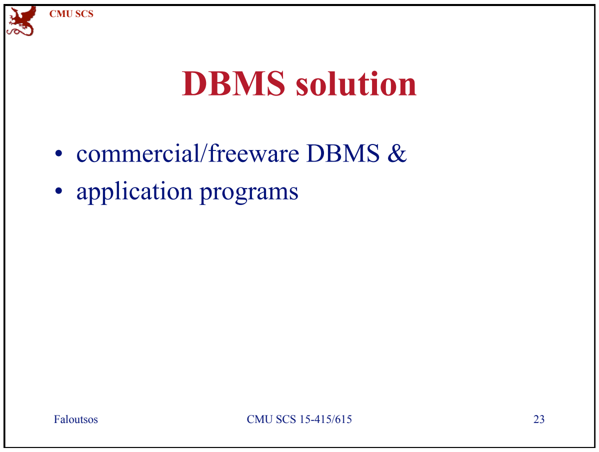
[why?

because of two main reasons:

- file-layout description is buried within the C programs and
- there is no support for transactions (concurrency and recovery)

]

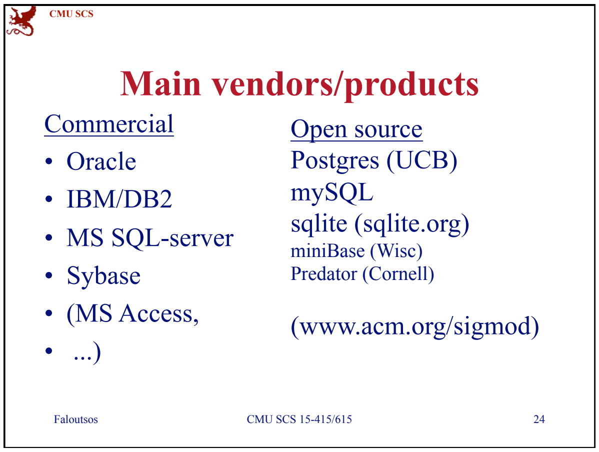
DBMSs handle exactly these two problems



CMU SCS

DBMS solution

- commercial/freeware DBMS &
- application programs



CMU SCS

Main vendors/products

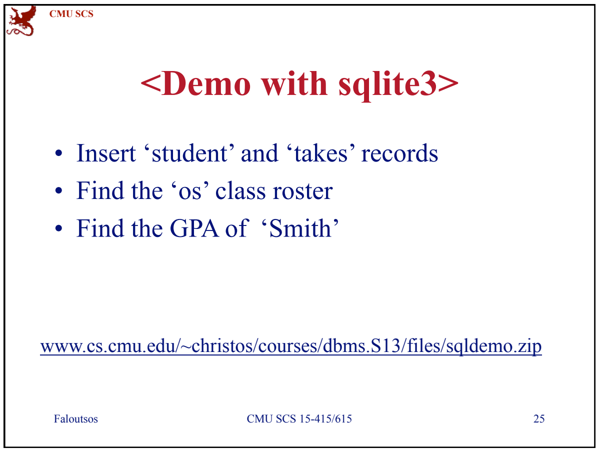
Commercial

- 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)

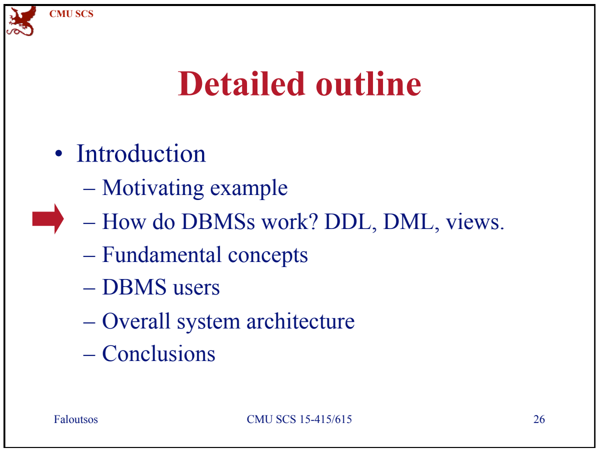


CMU SCS

<Demo with sqlite3>

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

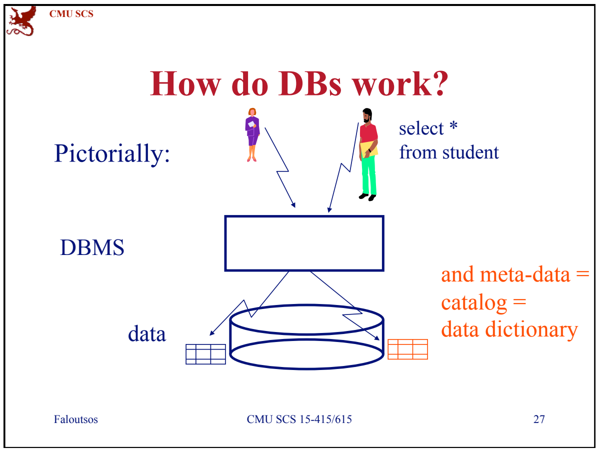
www.cs.cmu.edu/~christos/courses/dbms.S13/files/sqldemo.zip



CMU SCS

Detailed outline

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



CMU SCS

How do DBs work?

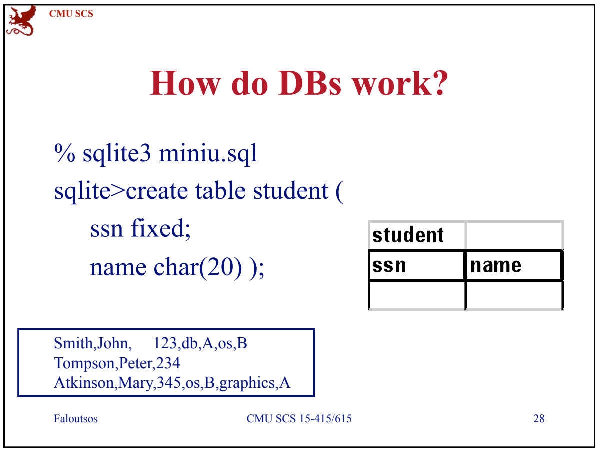
Pictorially:

select *
from student

DBMS

data

and meta-data =
catalog =
data dictionary

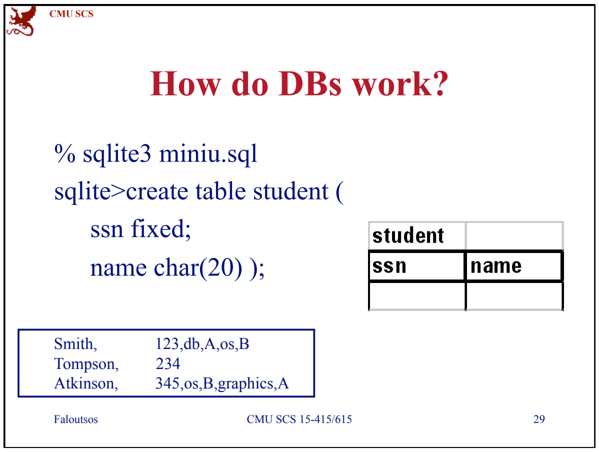


How do DBs work?

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

student	
ssn	name

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

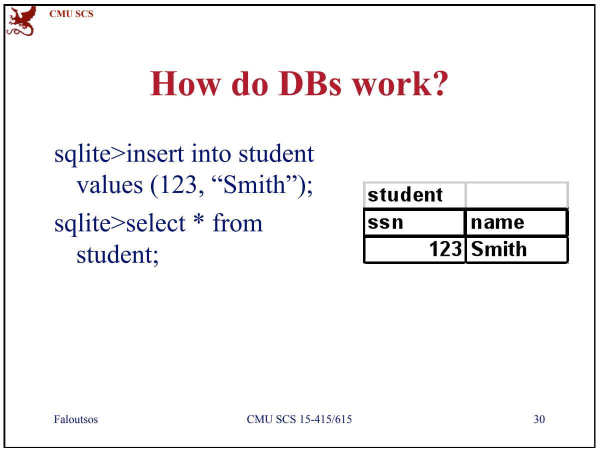


How do DBs work?

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

student	
ssn	name

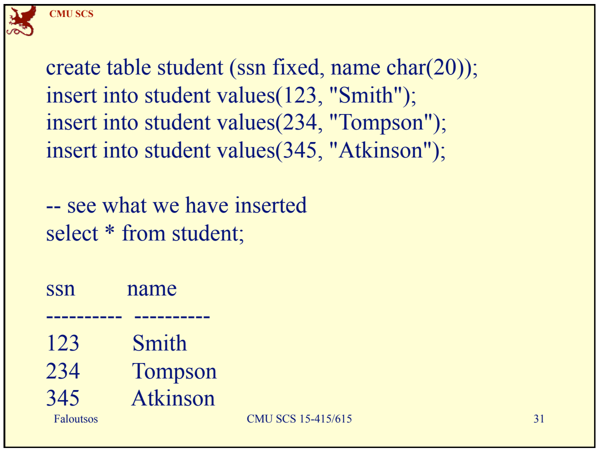
```
Smith,          123,db,A,os,B
Tompson,        234
Atkinson,       345,os,B,graphics,A
```



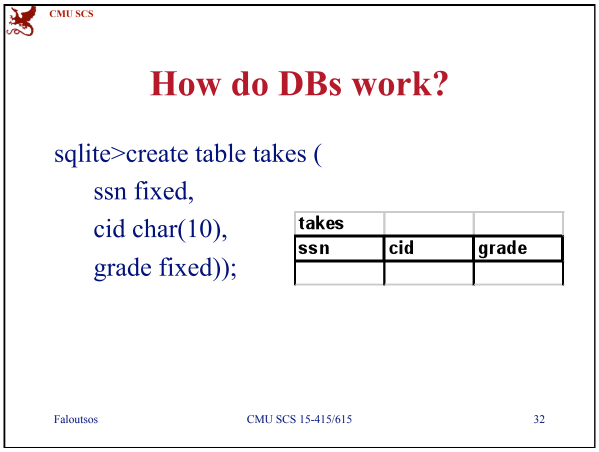
How do DBs work?

```
sqlite>insert into student
   values (123, “Smith”);
sqlite>select * from
   student;
```

student	
ssn	name
123	Smith



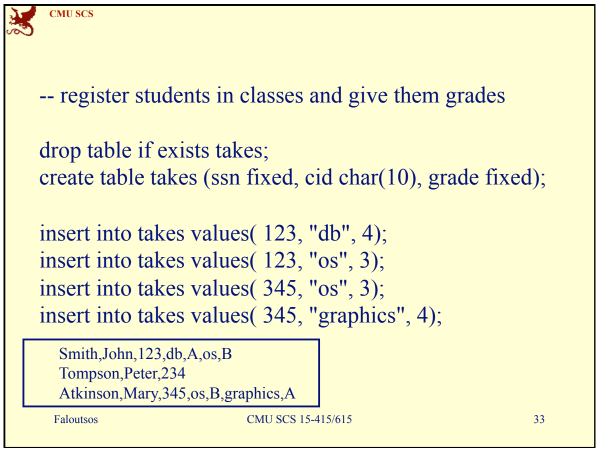
```
create table student (ssn fixed, name char(20));
insert into student values(123, "Smith");
insert into student values(234, "Tompson");
insert into student values(345, "Atkinson");

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

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



How do DBs work?

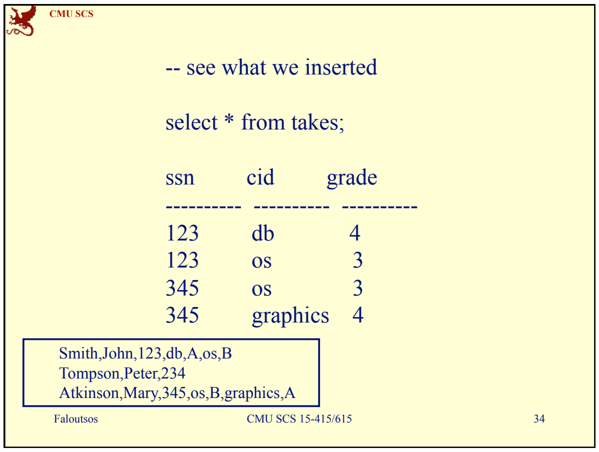
```
sqlite>create table takes (
     ssn fixed,
     cid char(10),
     grade fixed));
```

takes		
ssn	cid	grade



```
-- 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);
```

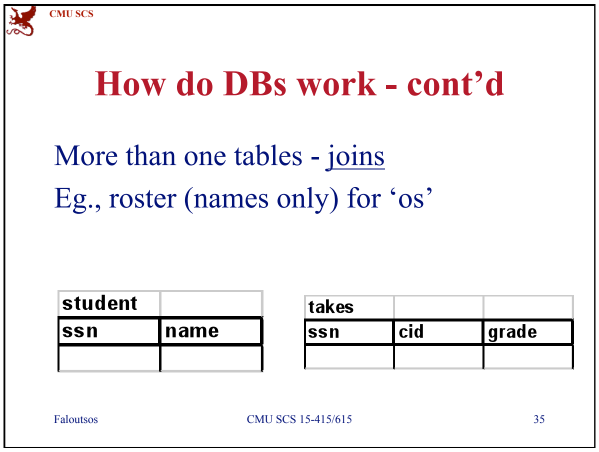
```
Smith,John,123,db,A,os,B
Tompson,Peter,234
Atkinson,Mary,345,os,B,graphics,A
```



CMU SCS

```
-- see what we inserted

select * from takes;

ssn         cid         grade
----------  ----------  ----------
123         db          4
123         os          3
345         os          3
345         graphics    4
```

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



CMU SCS

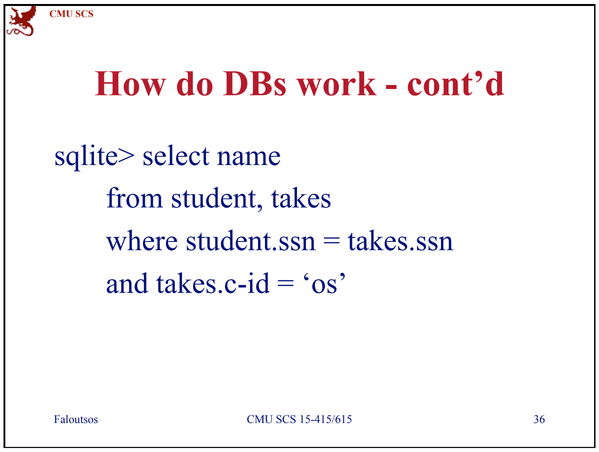
How do DBs work - cont’d

More than one tables - joins

Eg., roster (names only) for ‘os’

student	
ssn	name

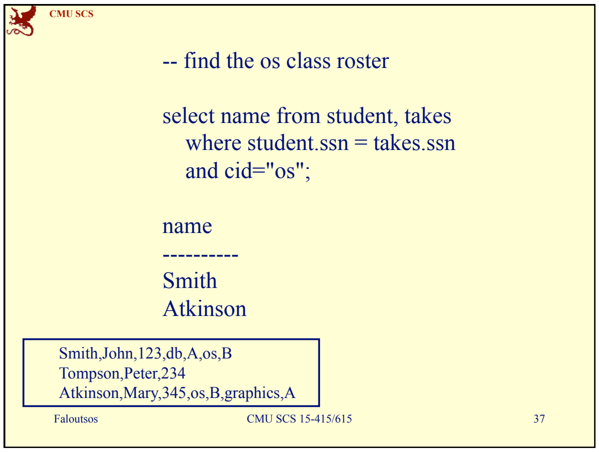
takes		
ssn	cid	grade



CMU SCS

How do DBs work - cont’d

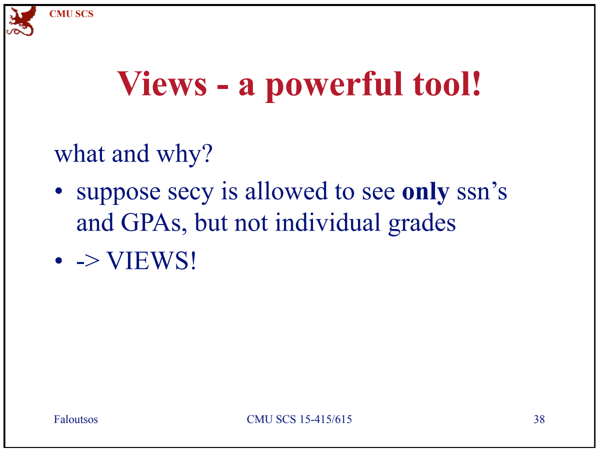
```
sqlite> select name
    from student, takes
    where student.ssn = takes.ssn
    and takes.c-id = ‘os’
```



CMU SCS

```
-- find the os class roster

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

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

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

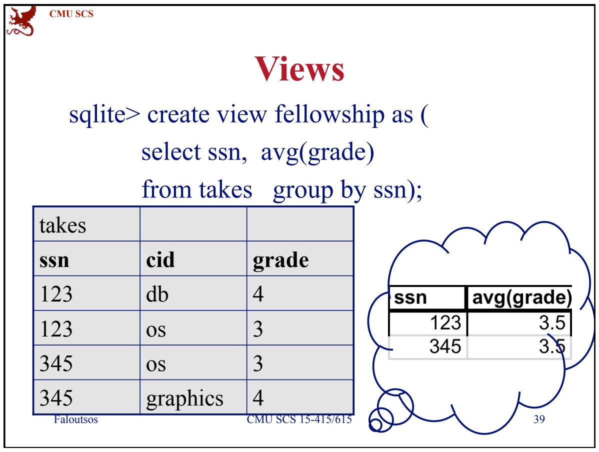


CMU SCS

Views - a powerful tool!

what and why?

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



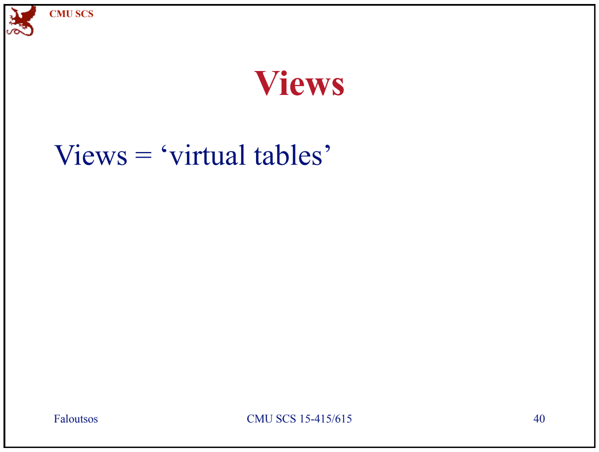
CMU SCS

Views

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

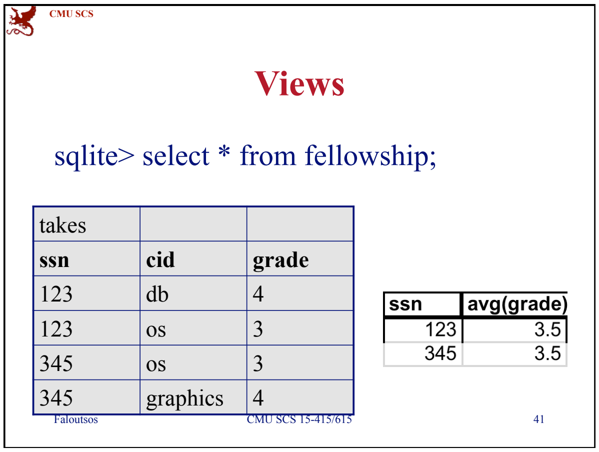
takes		
ssn	**cid**	**grade**
123	db	4
123	os	3
345	os	3
345	graphics	4

ssn	avg(grade)
123	3.5
345	3.5



Views

Views = ‘virtual tables’

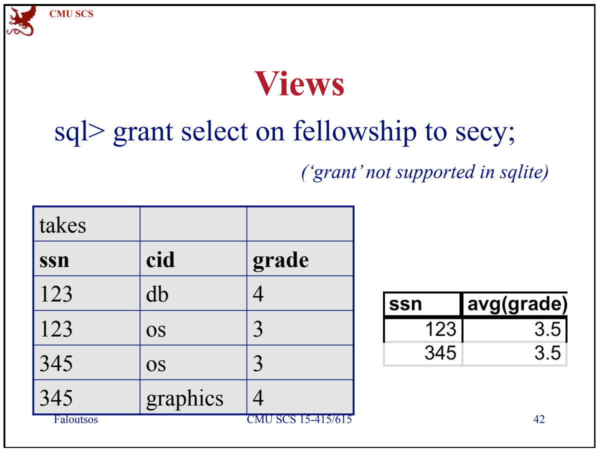


Views

sqlite> select * from fellowship;

takes		
ssn	**cid**	**grade**
123	db	4
123	os	3
345	os	3
345	graphics	4

ssn	avg(grade)
123	3.5
345	3.5



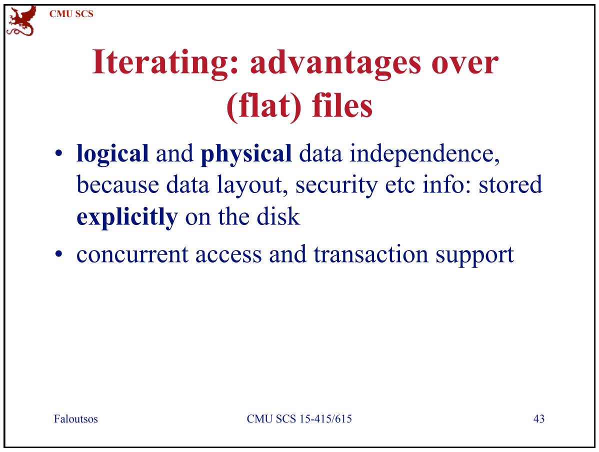
Views

sql> grant select on fellowship to secy;

(‘grant’ not supported in sqlite)

takes		
ssn	**cid**	**grade**
123	db	4
123	os	3
345	os	3
345	graphics	4

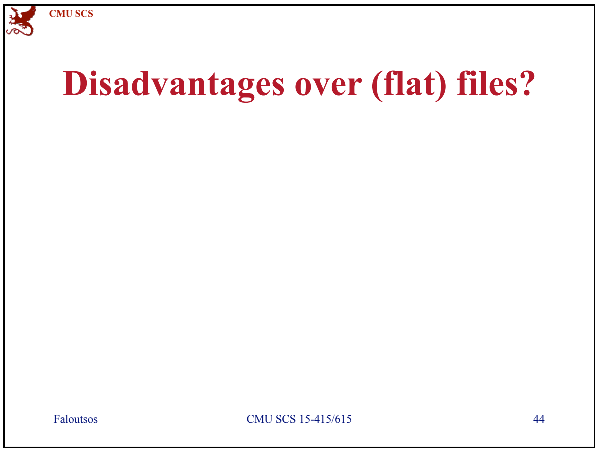
ssn	avg(grade)
123	3.5
345	3.5



CMU SCS

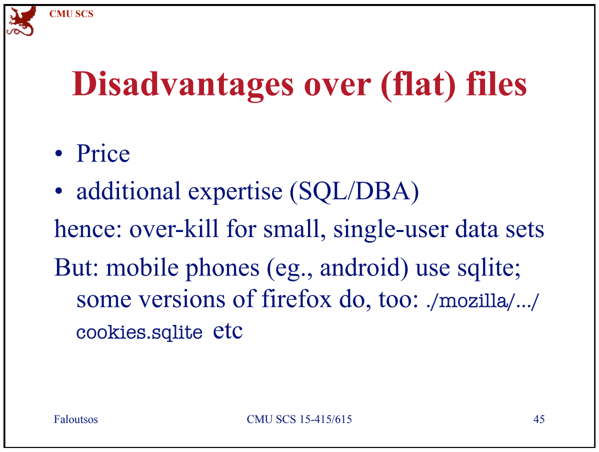
Iterating: advantages over (flat) files

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



CMU SCS

Disadvantages over (flat) files?



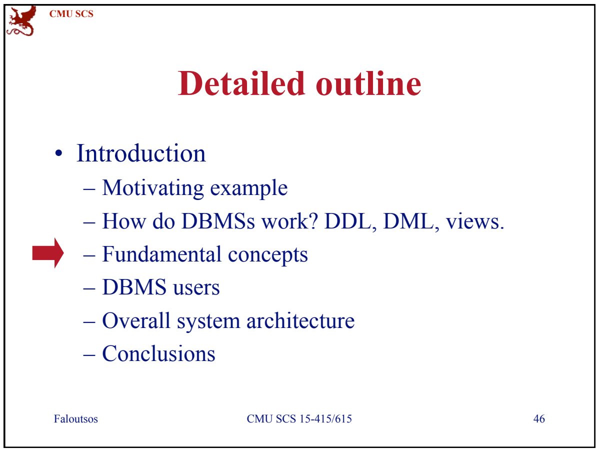
CMU SCS

Disadvantages over (flat) files

- Price
- additional expertise (SQL/DBA)

hence: over-kill for small, single-user data sets

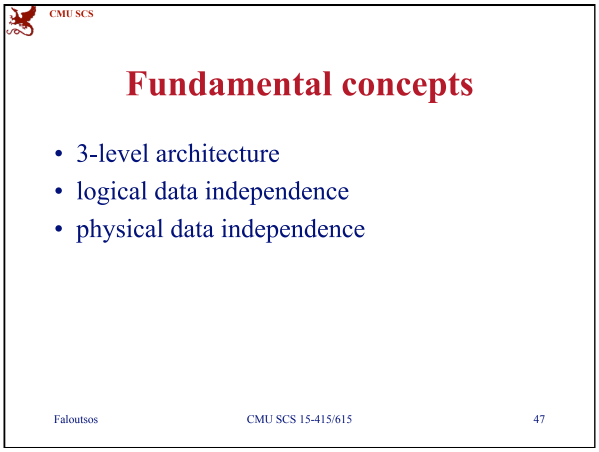
But: mobile phones (eg., android) use sqlite; some versions of firefox do, too: ./mozilla/…/cookies.sqlite etc



CMU SCS

Detailed outline

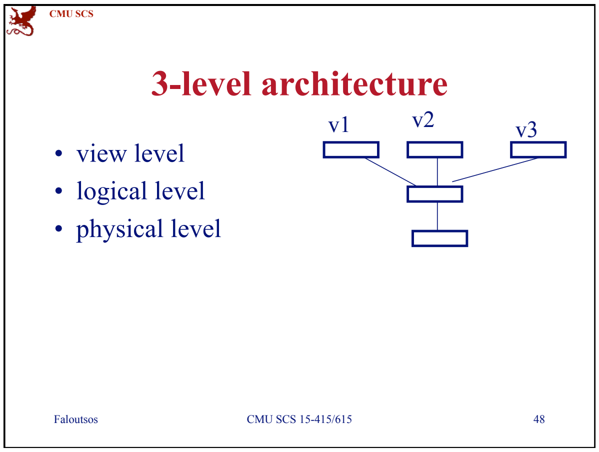
- Introduction
  - Motivating example
  - How do DBMSs work? DDL, DML, views.
  - ➡ Fundamental concepts
  - DBMS users
  - Overall system architecture
  - Conclusions



CMU SCS

Fundamental concepts

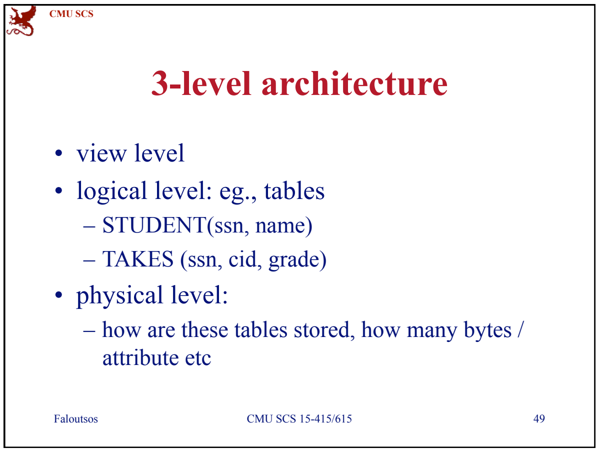
- 3-level architecture
- logical data independence
- physical data independence



CMU SCS

3-level architecture

- view level
- logical level
- physical level

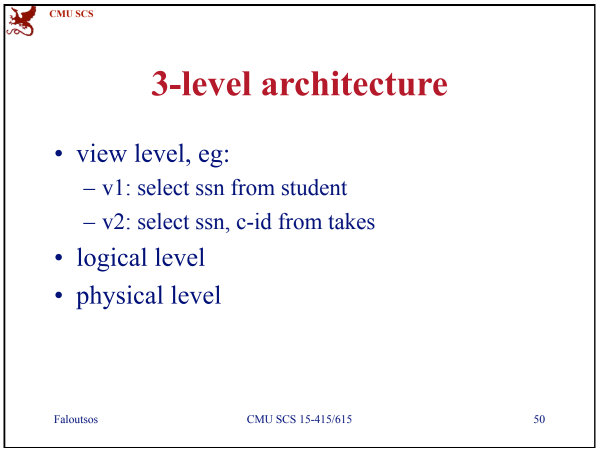


CMU SCS

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

Faloutsos CMU SCS 15-415/615 49

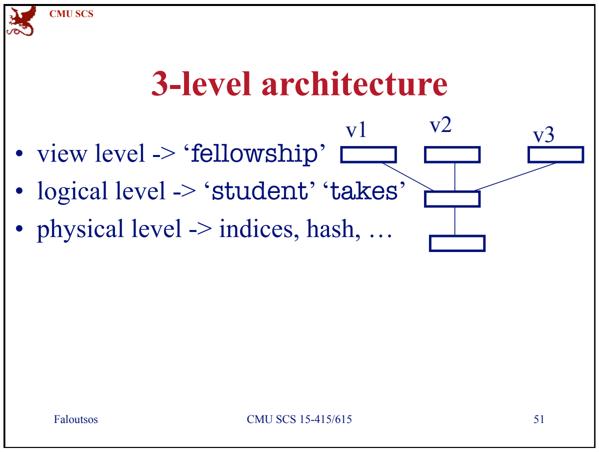


CMU SCS

3-level architecture

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

Faloutsos CMU SCS 15-415/615 50

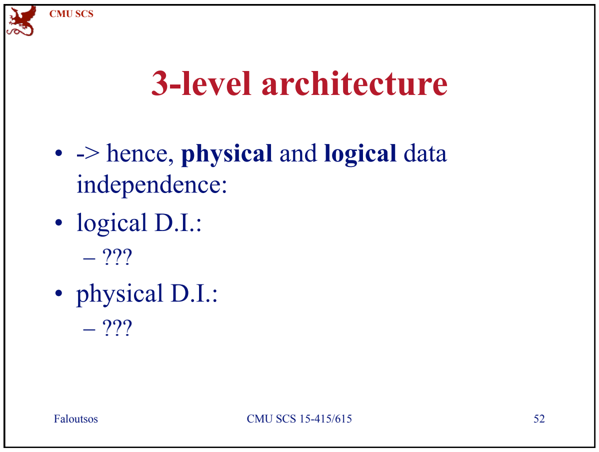


CMU SCS

3-level architecture

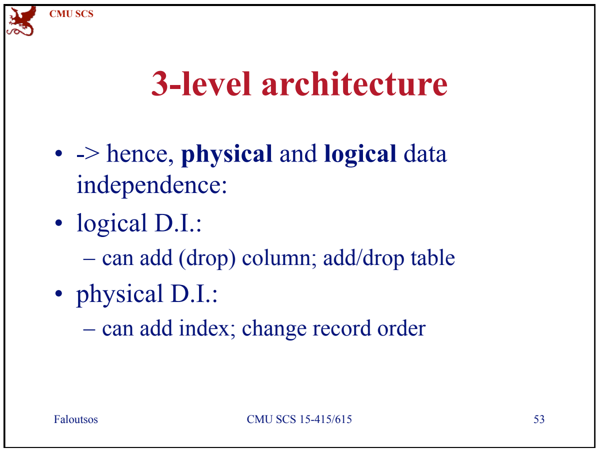
- view level -> 'fellowship'
- logical level -> 'student' 'takes'
- physical level -> indices, hash, …

Faloutsos CMU SCS 15-415/615 51



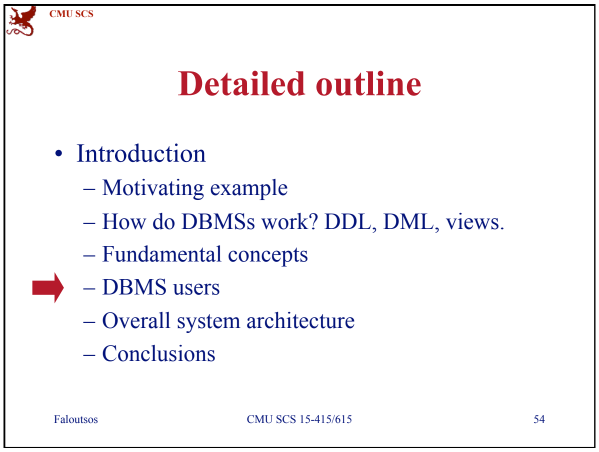
3-level architecture

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



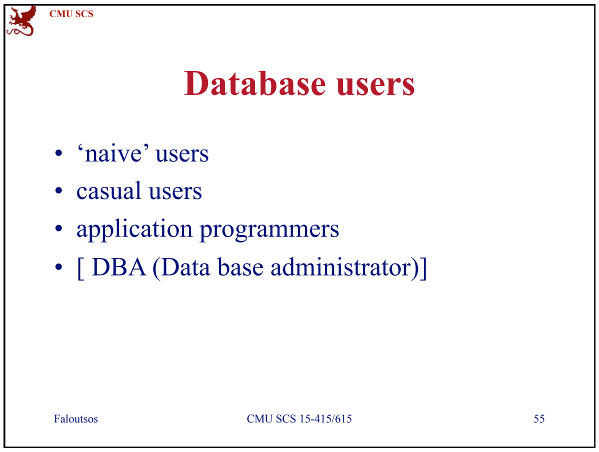
3-level architecture

- -> hence, **physical** and **logical** data independence:
- logical D.I.:
  - can add (drop) column; add/drop table
- physical D.I.:
  - can add index; change record order



Detailed outline

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

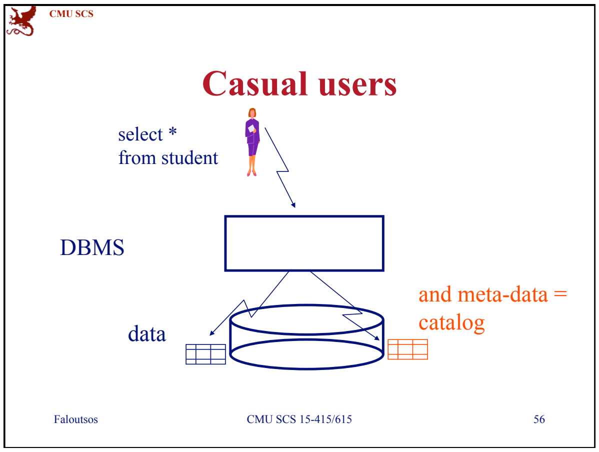


CMU SCS

Database users

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

Faloutsos CMU SCS 15-415/615 55



CMU SCS

Casual users

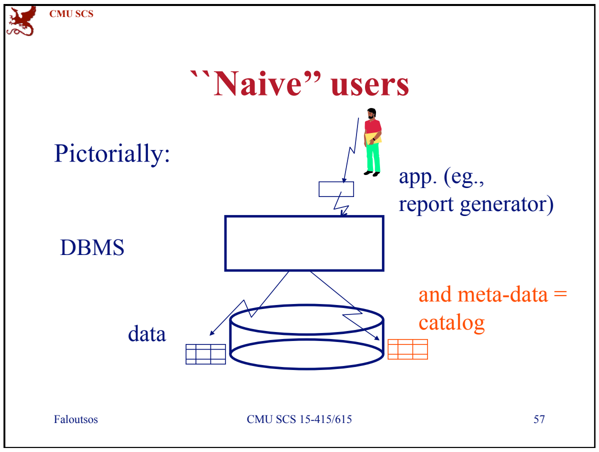
select *
from student

DBMS

data

and meta-data =
catalog

Faloutsos CMU SCS 15-415/615 56



CMU SCS

``Naive” users

Pictorially:

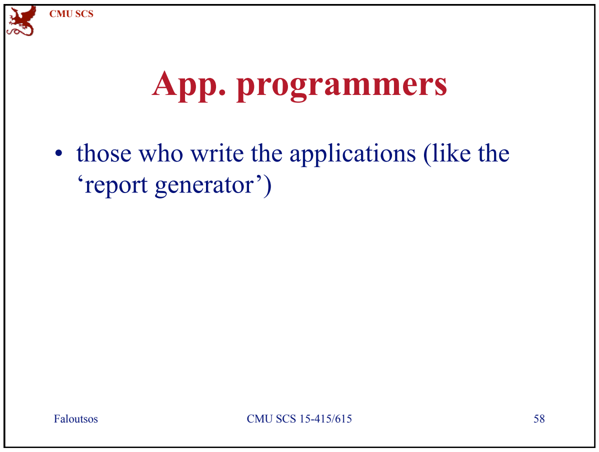
app. (eg.,
report generator)

DBMS

data

and meta-data =
catalog

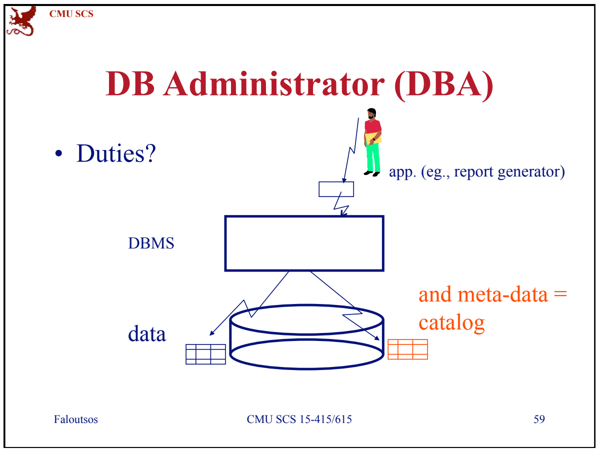
Faloutsos CMU SCS 15-415/615 57



CMU SCS

App. programmers

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



CMU SCS

DB Administrator (DBA)

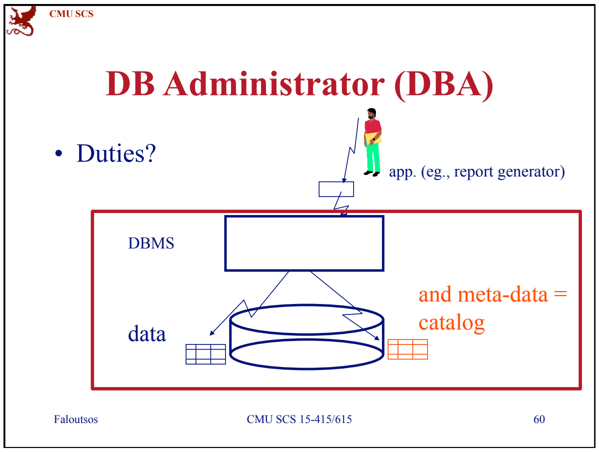
- Duties?

app. (eg., report generator)

DBMS

data

and meta-data = catalog



CMU SCS

DB Administrator (DBA)

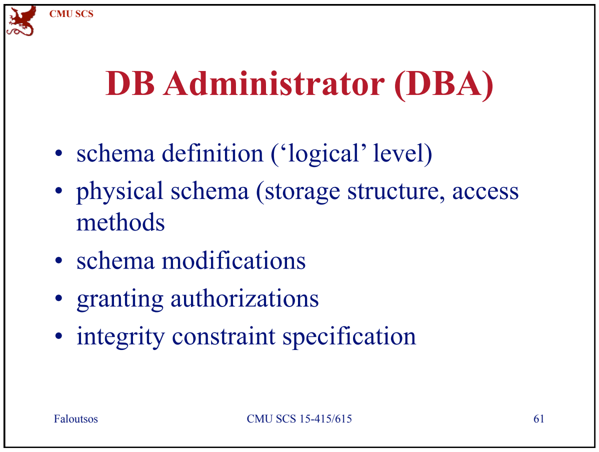
- Duties?

app. (eg., report generator)

DBMS

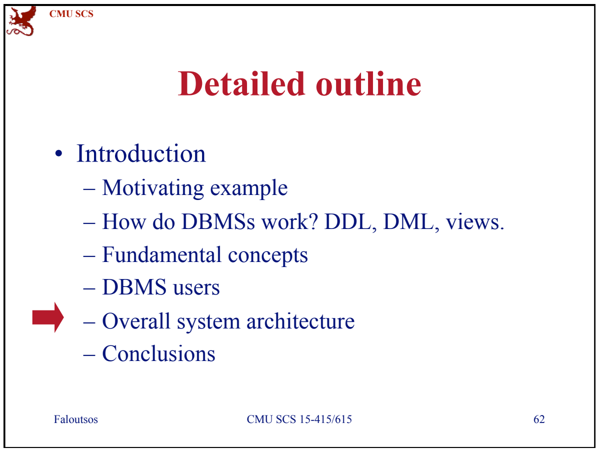
data

and meta-data = catalog



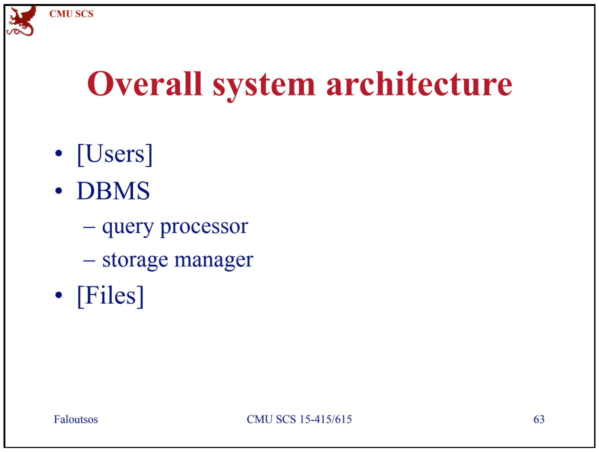
DB Administrator (DBA)

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



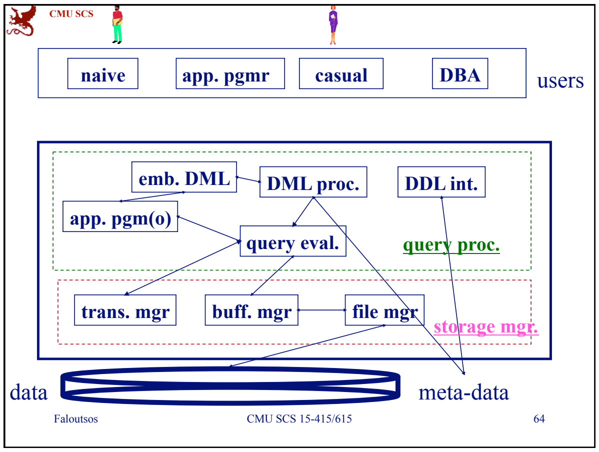
Detailed outline

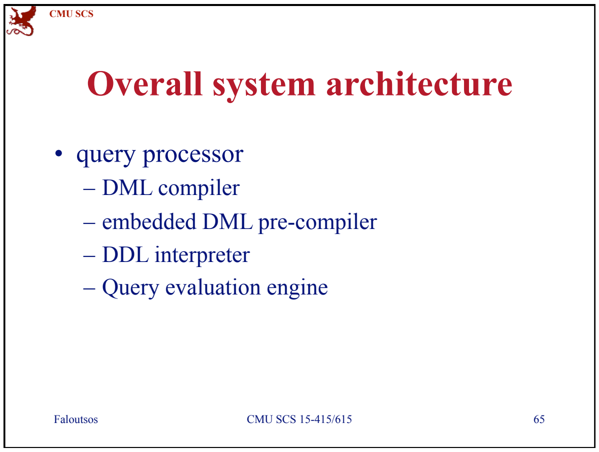
- Introduction
  - Motivating example
  - How do DBMSs work? DDL, DML, views.
  - Fundamental concepts
  - DBMS users
  - ➡ Overall system architecture
  - Conclusions



Overall system architecture

- [Users]
- DBMS
  - query processor
  - storage manager
- [Files]



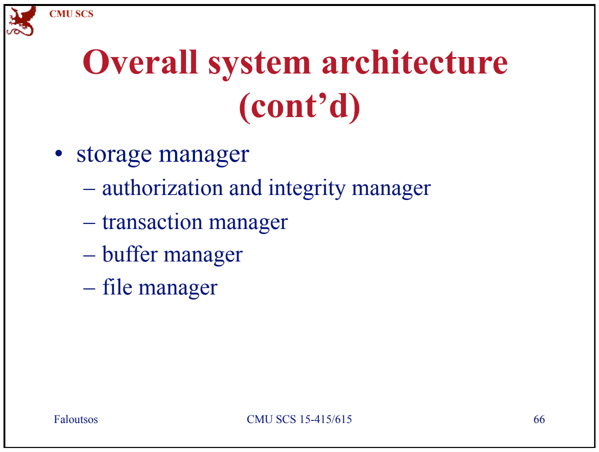


CMU SCS

Overall system architecture

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

Faloutsos CMU SCS 15-415/615 65



CMU SCS

Overall system architecture (cont'd)

- storage manager
  - authorization and integrity manager
  - transaction manager
  - buffer manager
  - file manager

Faloutsos CMU SCS 15-415/615 66

CMU SCS

Overall system architecture (cont'd)

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

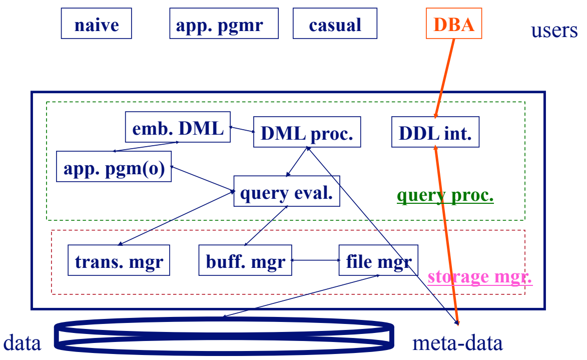
CMU SCS

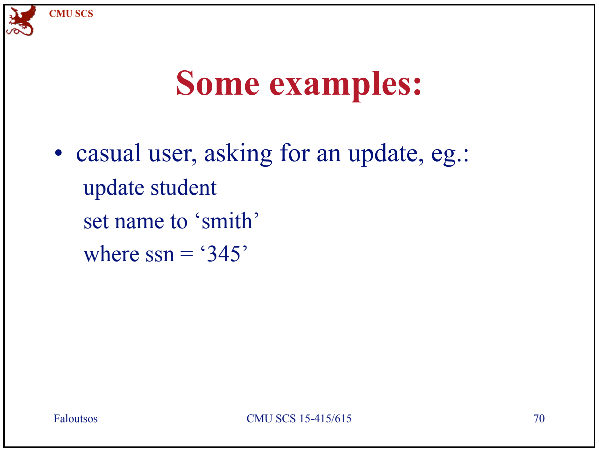
Some examples:

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

  create table student ...

CMU SCS

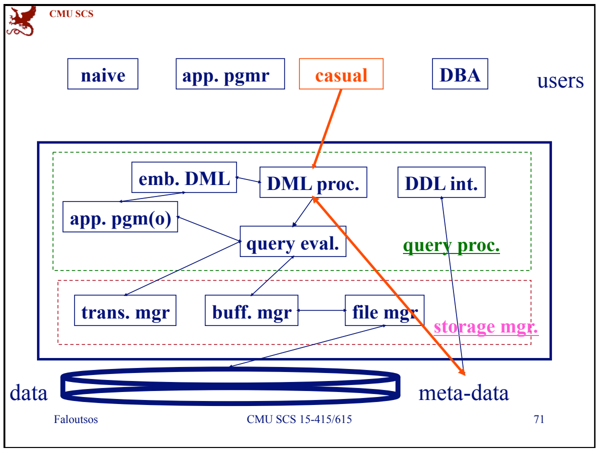




Some examples:

- casual user, asking for an update, eg.:

  update student
  set name to 'smith'
  where ssn = '345'



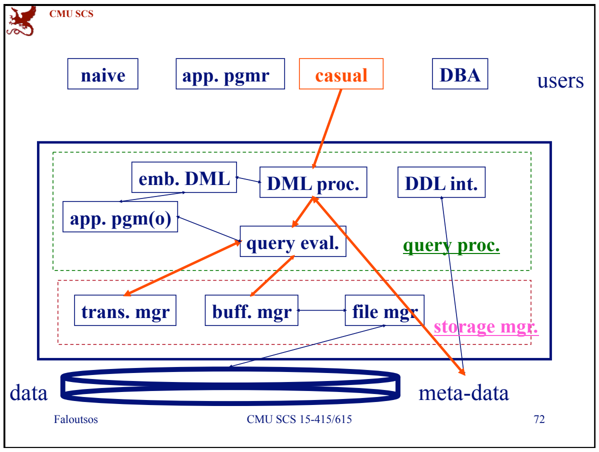
naive app. pgmr casual DBA users

emb. DML DML proc. DDL int.

app. pgm(o) query eval. query proc.

trans. mgr buff. mgr file mgr storage mgr.

data meta-data



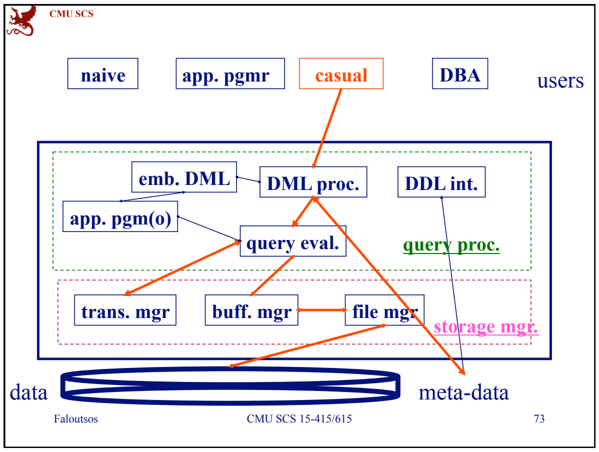
naive app. pgmr casual DBA users

emb. DML DML proc. DDL int.

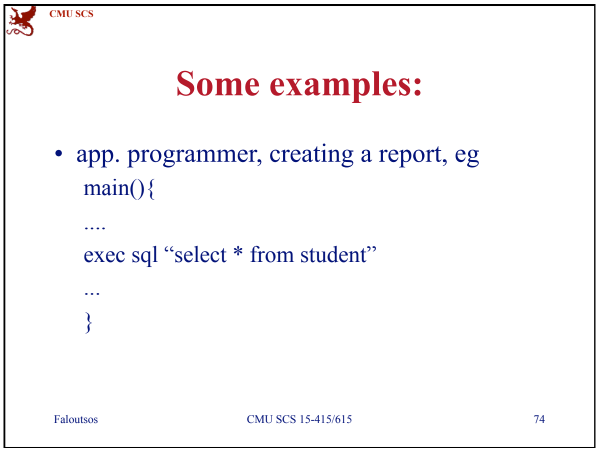
app. pgm(o) query eval. query proc.

trans. mgr buff. mgr file mgr storage mgr.

data meta-data



Faloutsos CMU SCS 15-415/615 73



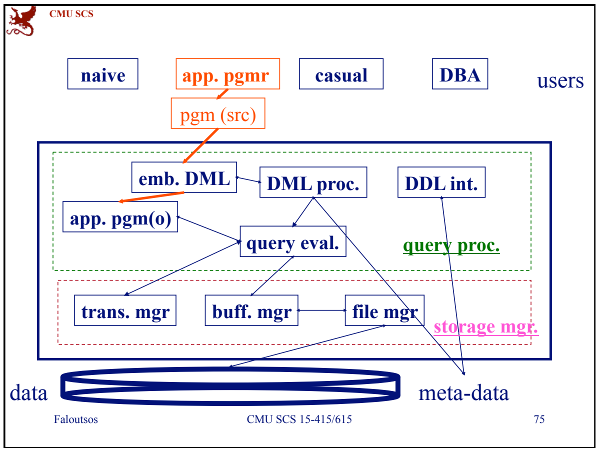
CMU SCS

Some examples:

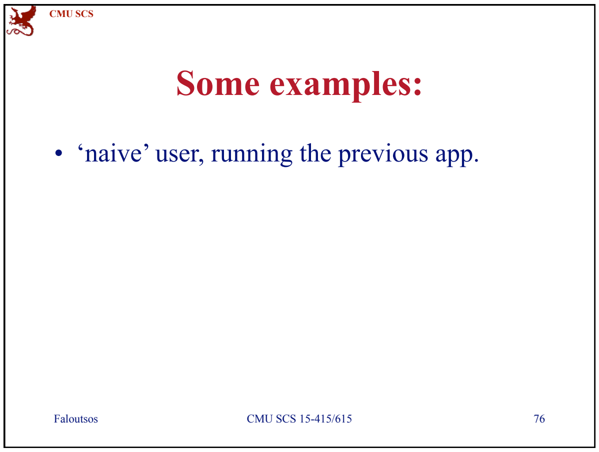
- app. programmer, creating a report, eg

```
main(){
....
exec sql “select * from student”
...
}
```

Faloutsos CMU SCS 15-415/615 74



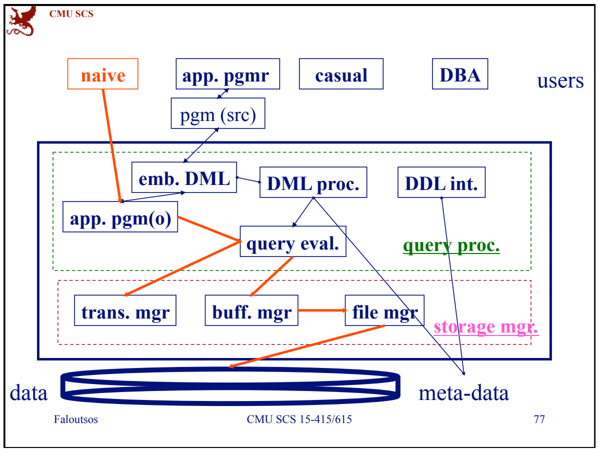
Faloutsos CMU SCS 15-415/615 75



CMU SCS

Some examples:

- ‘naive’ user, running the previous app.



CMU SCS

naive

app. pgmr

casual

DBA

users

pgm (src)

emb. DML

DML proc.

DDL int.

app. pgm(o)

query eval.

query proc.

trans. mgr

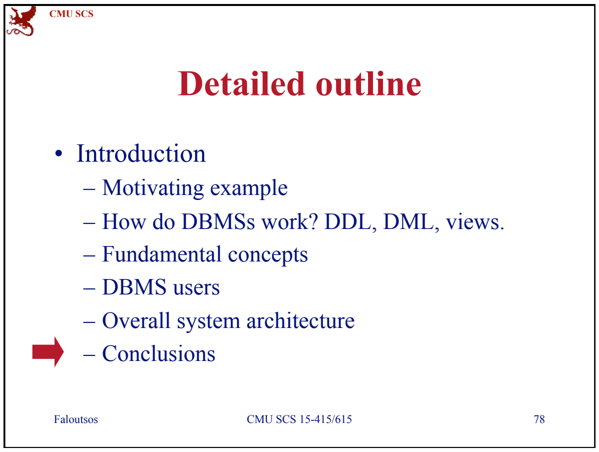
buff. mgr

file mgr

storage mgr.

data

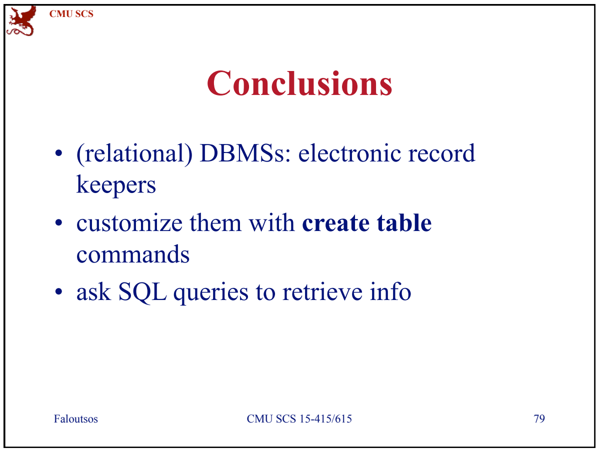
meta-data



CMU SCS

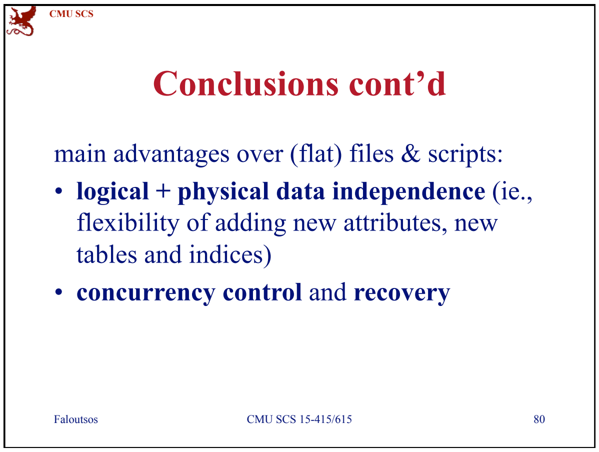
Detailed outline

- 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**