Contents

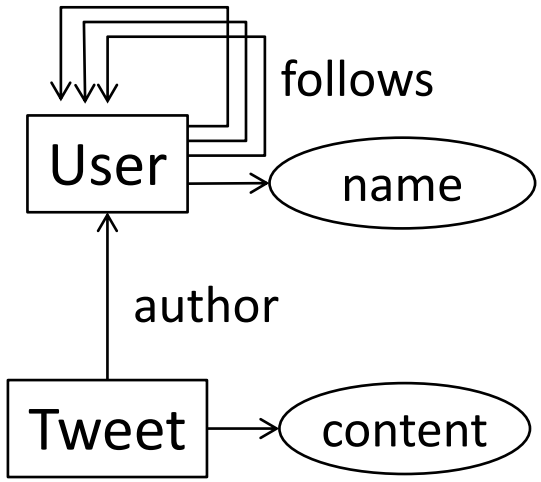
- Introduction and fundamentals
- Introduction to SQL
- Entity-relationship model
- Relational model
- Relational design: normal forms
- Queries
  - Relational calculus
  - Relational algebra
- Database implementation
  - Physical structure: fields and records
  - Indexing
    - Simple indices
    - B trees

A first informal example

- Twitter data
  - Users, follow relations, tweets…
- Queries and operations
  - View a user's timeline, follow someone, search for tweets…

In C – Data structures

```
struct User {
        char *name;
        User *follows[5000];
        int nfollows;
};
struct Tweet {
        char *content;
        struct User *author;
};
// Plus some structure to store all users
// and tweets, etc.
// ...
```



In C – Data creation

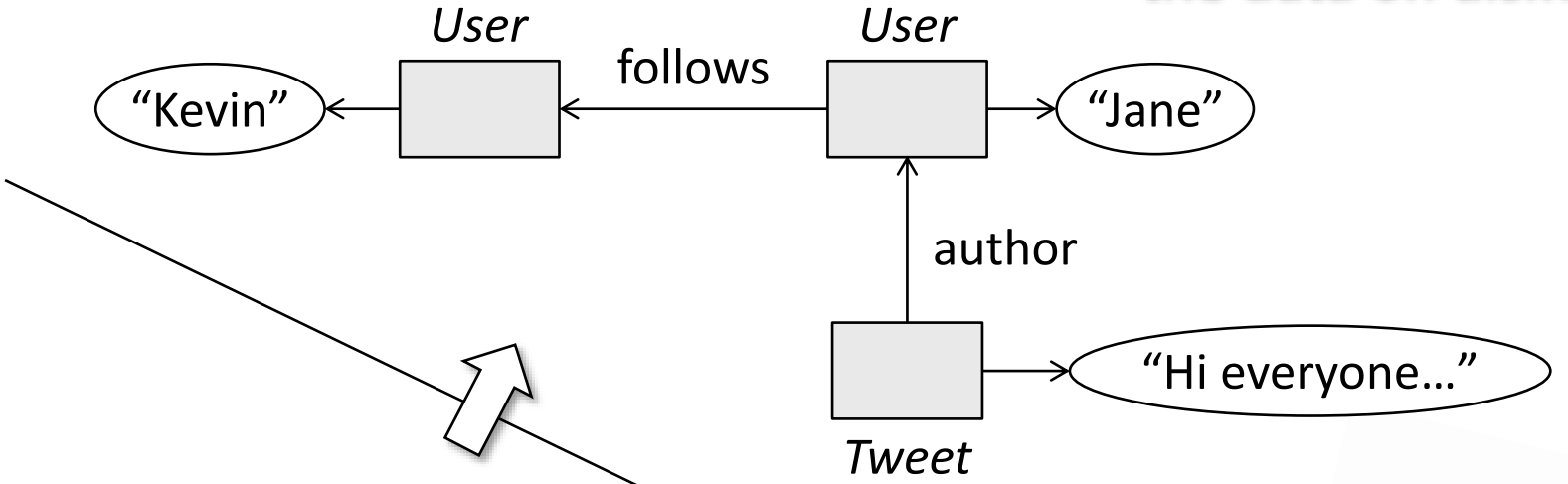
```
User *create_user (char *name) {
      User *u = (User*) malloc(sizeof (User*));
      u->name = name;
      u->nfollows = 0;
      return u;
}
void add_follows (User *u, User *v) {
      u->follows[u->nfollows++] = v;
}
Tweet *create_tweet (User *u, char *text) {
      Tweet *t = (Tweet*) malloc(sizeof(Tweet*));
      t->content = text;
      t->author = u;
      return t;
}
```

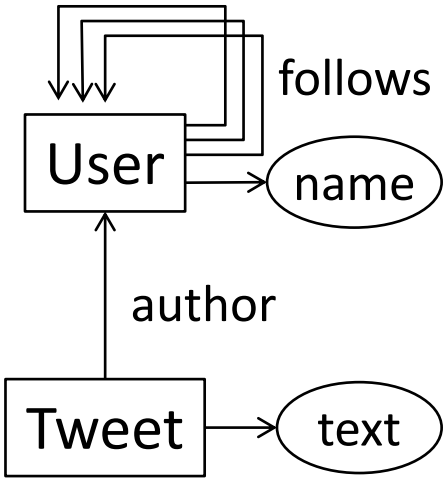
In C – Data creation (cont)

```
void main () {
   User *jane = create_user("Jane");
   User *kevin = create_user("Kevin");
   add_follows(jave, kevin);
   create_tweet(jane, "Hi everyone I just signed up");
   // ...
}
```

Data in RAM

What if we want to store the data on disk?



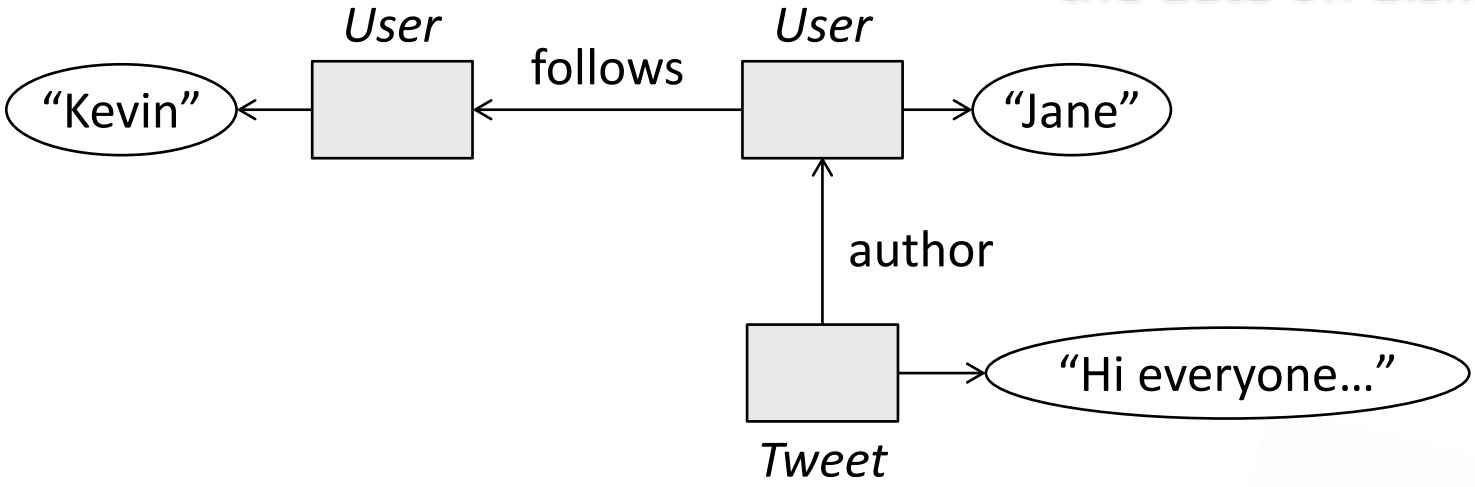


Data (memory)

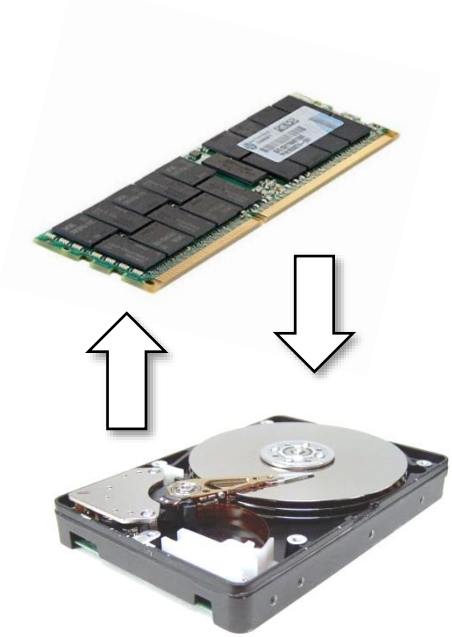
Data schemas (design)

Data in RAM

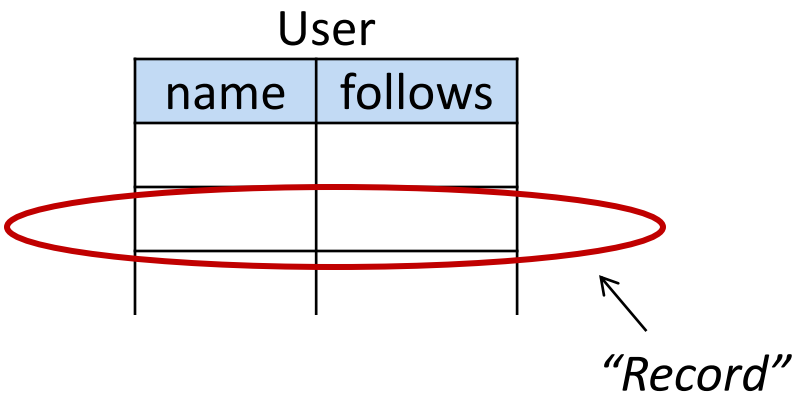
What if we want to store the data on disk?

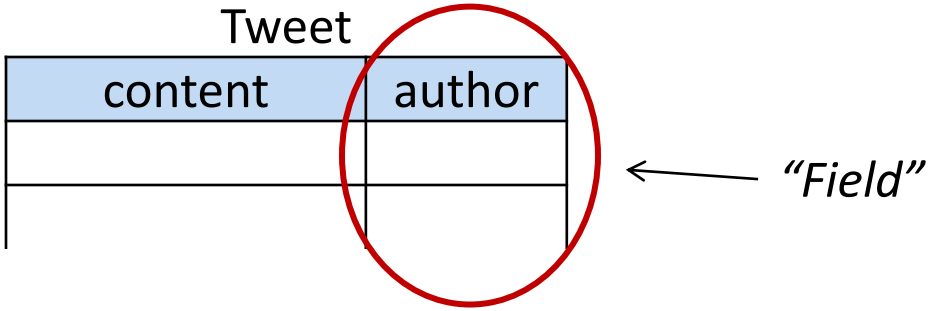


- Disk ⇒ Data in files
- What structure? → struct ~ tables!
- And pointers?
  We wish to be able to recover the structures in RAM, or even port to another machine



In tables – Data structures





Why tables?

- It is a very general structure, suits well many domains
- Suits well physical storage
- Suits well our way of thinking

In tables – Data creation

User

name	follows
“Jane”	?
“Kevin”	
…	

Tweet

content	author
“Hi everyone…”	?
…	

How are data physically stored?

- In a file “one after another” ☺
- We need to define how to separate data parts (rows and columns)
- Optimized access and update techniques are needed
- We will study this later…

In tables – Data creation

How to store “pointers”?

User

name	follows
“Jane”	?
“Kevin”	
…	

Tweet

content	author
“Hi everyone…”	?
…	

In tables – Data creation

How to store “pointers”?

“Primary key”

name	follows
“Jane”	?
“Kevin”	
…	

In a real case, it would be better to use a field such as the email or an internal ID

Tweet

content	author
“Hi everyone…”	“Jane”
…	

“Foreign key”

A dedicated field plays the part of pointer

In tables – Data creation

How to store “pointers”?

And arrays of pointers?

User

name	follows
“Jane”	?
“Kevin”	
…	

Tweet

content	author
“Hi everyone…”	“Jane”
…	

In tables – Data creation

How to store “pointers”?

And arrays of pointers?

“Primary key”

name	follows
“Jane”	
“Kevin”	
…	

“Foreign key”

Follows

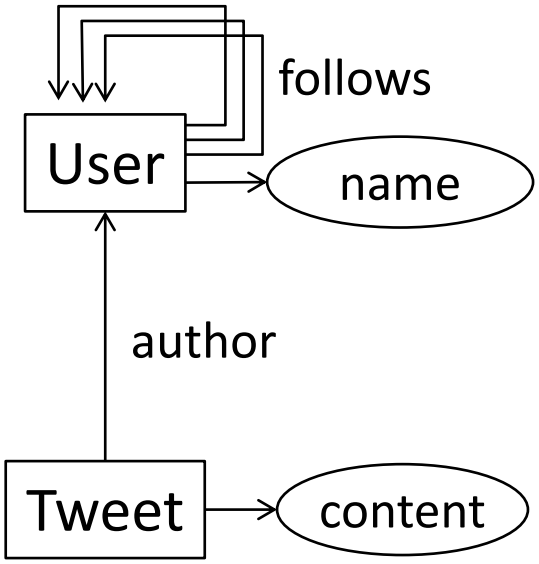
user1	user2
“Jane”	“Kevin”
“Jane”	…
…	…

“Foreign key”

Tweet

content	author
“Hi everyone…”	“Jane”
…	

In tables – Data design



User

name
“Jane”
“Kevin”
…

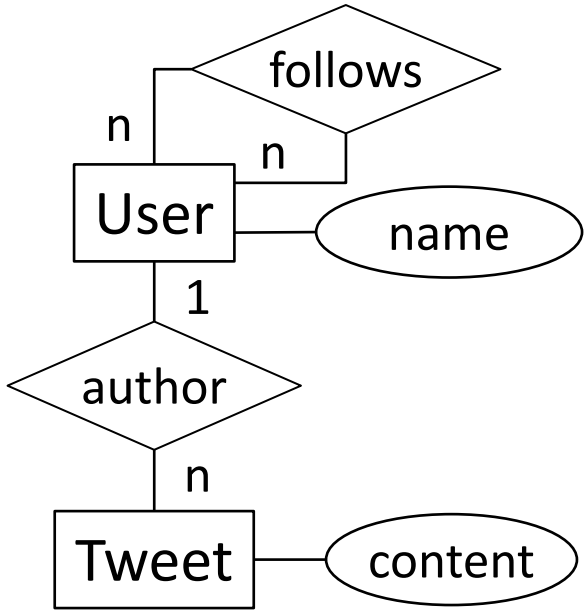
Follows

user1	user2
“Jane”	“Kevin”
“Jane”	…
…	…

Tweet

content	author
“Hi everyone…”	“Jane”
…	…

In tables – Data design



User

name
“Jane”
“Kevin”
…

Follows

user1	user2
“Jane”	“Kevin”
“Jane”	…
…	…

Tweet

content	author
“Hi everyone…”	“Jane”
…	…

ER model

- Entity
- Relationship
- Property
- …

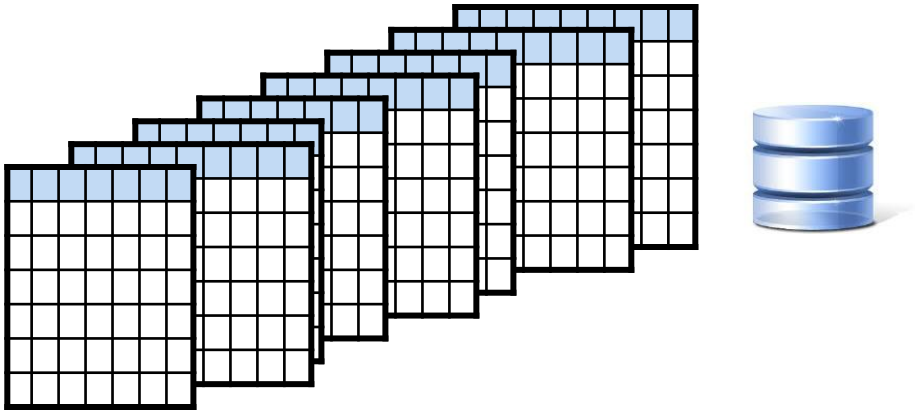
Databases

Simplifying a bit…

A **database** is

a set of tables

on disk



How do we implement and manage tables on disk?

- Do we implement it all from scratch? Usually not…
- We use a **Database Management System** (DBMS)
- And the **SQL** language

DBMS and SQL

- SQL provides syntax for:
  - Defining table structures
  - Introducing, deleting and modifying data in tables
  - Running both simple and complex queries
- A DBMS provides:
  - An SQL engine
  - A user interface to apply operations on databases (create, open, browse, update, etc.) interactively instead of using SQL
  - DB administration functionalities: users, permissions, configuration, etc.

Example in PostgreSQL…

SQL examples – Table creation

```
create table TwitterUser (   -- "User" is reserved in SQL
  email varchar primary key, -- Slight variation...
  name varchar
);
create table Follows (
  user1 varchar references TwitterUser(email),
  user2 varchar references TwitterUser(email)
);
create table Tweet (
  author varchar references TwitterUser(email),
  content text
);
```

SQL examples – Data insertion

```
-- Data insertion in the three tables
insert into TwitterUser values
  ('jane@gmail.com', 'Jane'),
  ('kevin@gmail.com', 'Kevin');

insert into Follows values
  ('jane@gmail.com', 'kevin@gmail.com');

insert into Tweet values
  ('jane@gmail.com ', 'Hi everyone I just signed up');
```

SQL examples – Queries

```
-- Users named "Jane"
select email, name from TwitterUser
where name = 'Jane'

-- Tweets posted by users named "Jane"
select content from Tweet, TwitterUser
where author = email and name = 'Jane'

-- User with most followers
select name, count(email) as nfollowers
from TwitterUser, Follows
where email = user2
group by email, name
order by nfollowers desc limit 1
```

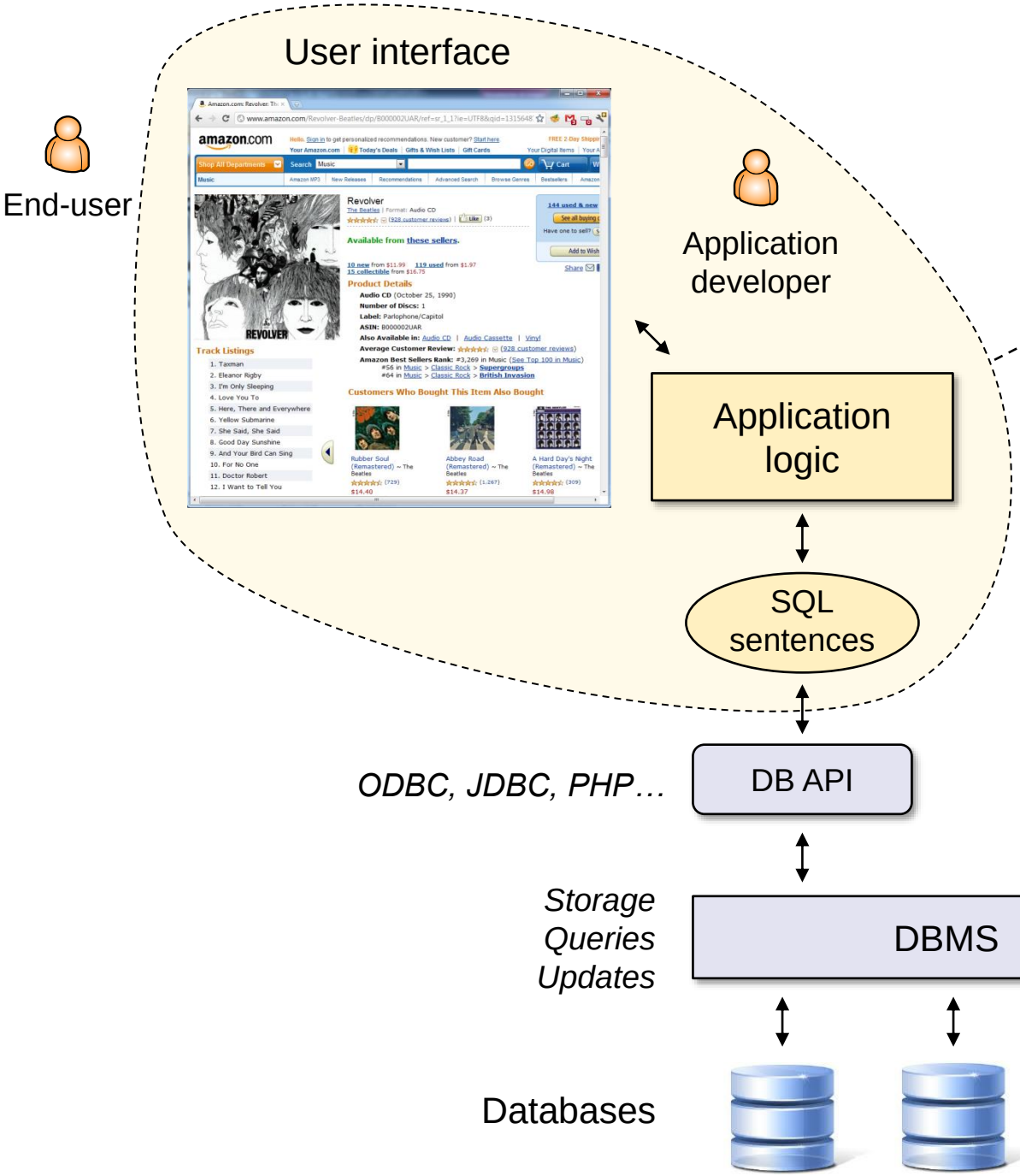
In C it would be something like…

```
User *getUser (User *users[], int nusers, char *name) {
  for (int i = 0; i < nusers; i++)
    if (strcpy(users[i]->name, name) == 0)
      return users[i];
  return NULL;
}
User *mostfollowed (User *users[], int nusers) {
  int nfollowers[nusers], max = 0;
  for (int i = 0; i < nusers; i++)
    for (int j = 0; j < users[i]->nfollows; j++)
      if (++nfollowers[users[i]->follows[j]] > nfollowers[max])
        max = users[i]->follows[j];
  return users[max];
}
```

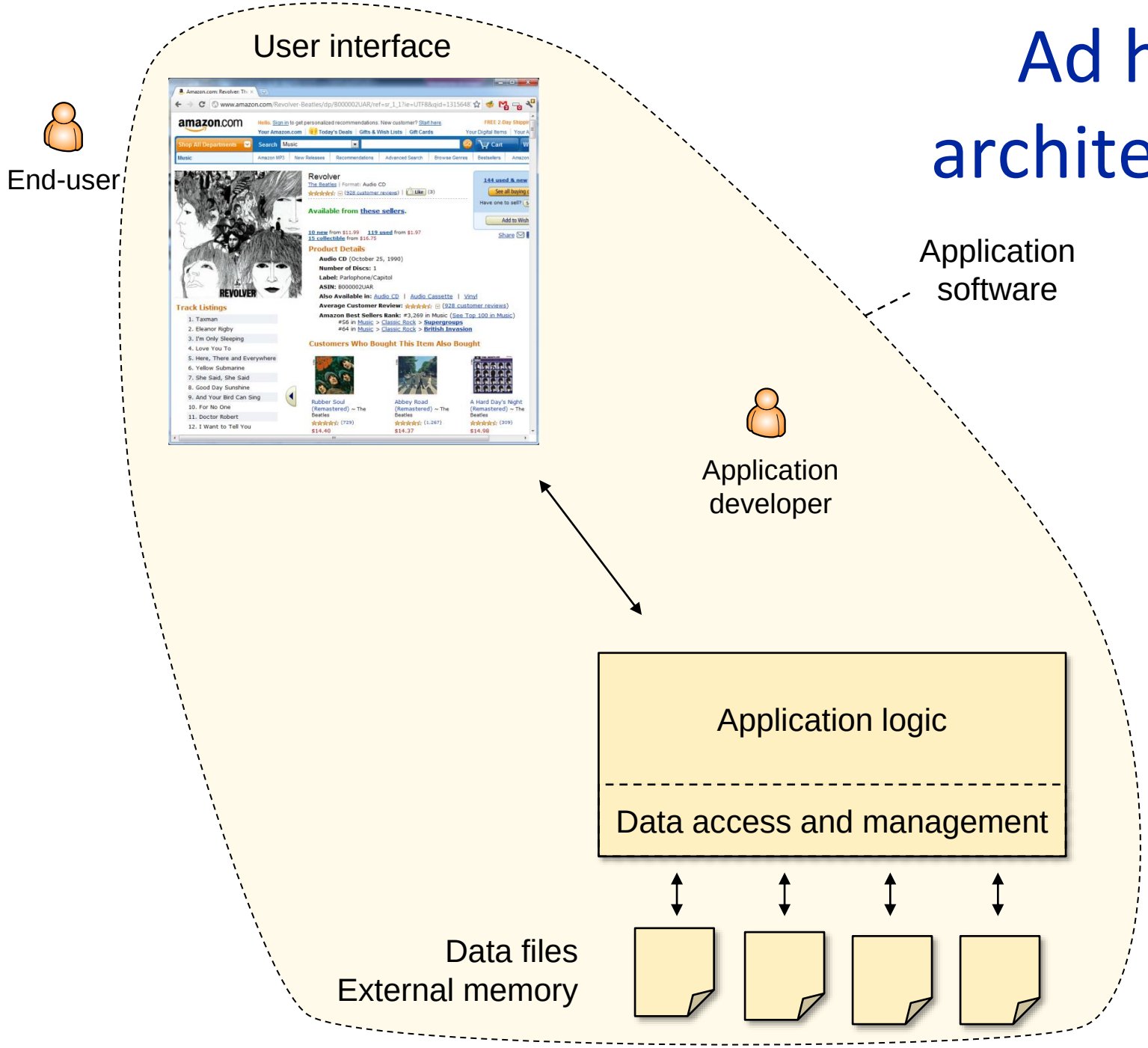
Database Management System (DBMS)

- Database management and access software
  - The development of a DBMS involves thousands of person · year
- SQL processor
  - Engine for running operations
  - Query optimizer
- Physical storage engine
- Administration tools
  - Creation and design of tables, users…
- APIs with C, Java, Python, PHP…

DB architecture



Ad hoc architecture



Roles in the use of a database

- End users
  - Interact with applications that access the DB
- Power users
  - Interact with the DB in SQL
- Application developers
  - Interact with the DB by writing programs
- Designers
  - Define the DB design
- Administrators
  - Maintain the DB design
  - Manage users and access permission
  - Manage update needs
- DBMS developers
  - Implement the lowest physical data access layer
  - Develop the software and tools that service all the above

Other levels of abstraction

In addition to SQL we will study…

- The formalisms DBs are based upon
  - Relational model (design primitives)
  - Normal forms (design quality)
  - Relational calculus and algebra (to build queries)
- Implementation techniques internal to a DBMS

Brief temporal perspective

1960's	First database notions
1970	Relational model proposed (E. F. Codd, CACM)
1974	First DBMS at MIT (RDMS)
	SQL at IBM (D. D. Chamberlin & R. F. Boyce)
1976	Entity / Relationship model
1979	Oracle
1980	dBase II
1983	IBM DB2
1984	FoxPro
Mid 80's	DB technology deployment
1987	SAP Sybase
1989	MS SQL Server
1992	MS Access
1994	MySQL
1995	PostgreSQL
Mid 90's	Object-oriented DBs
2000's	XML DBs, distributed DBs, Big Data…