By *Marina Barsky*

CSC 343

Introduction to databases

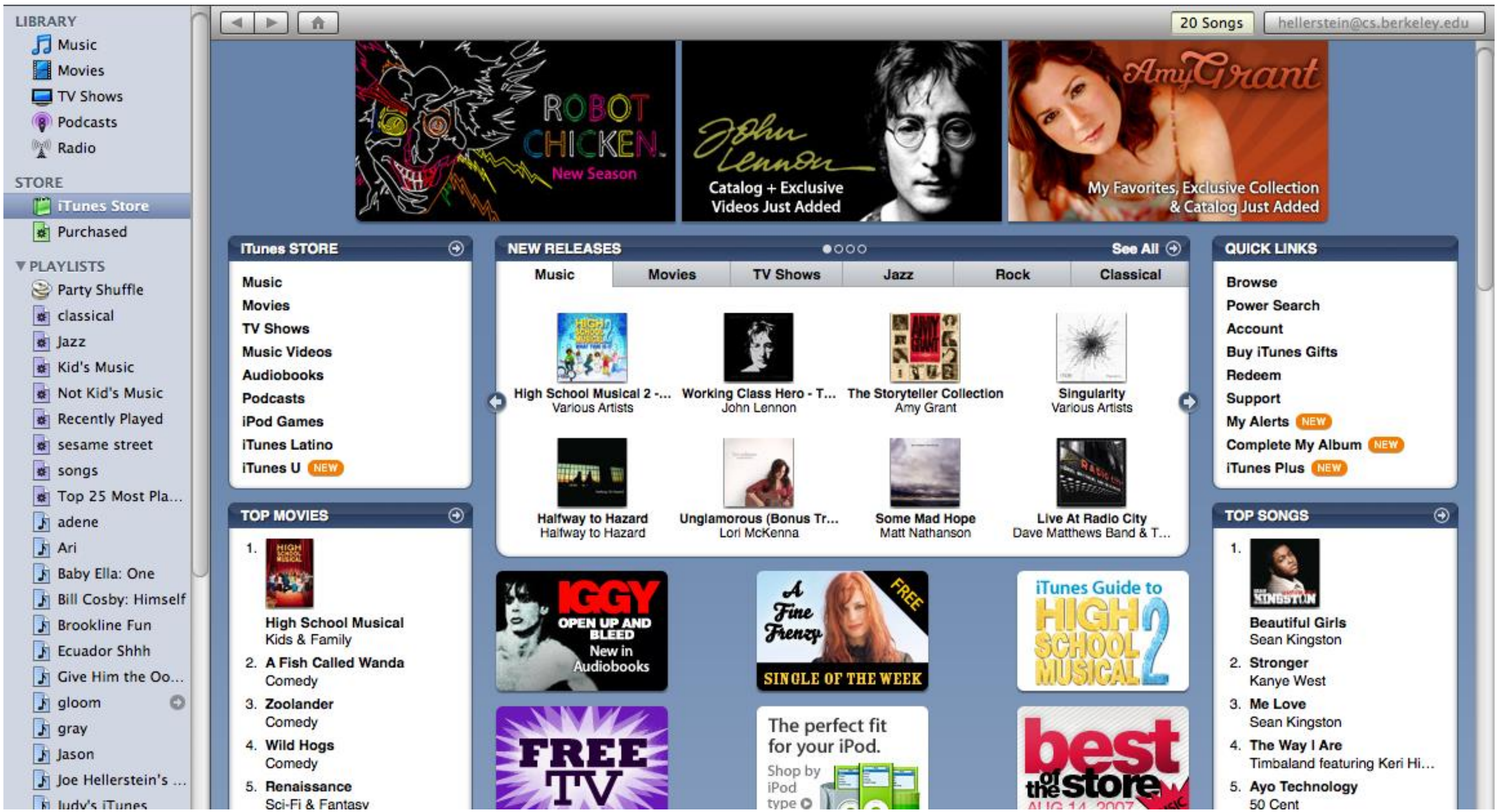
Summer 2016

http://www.cdf.toronto.edu/~csc343h/summer/

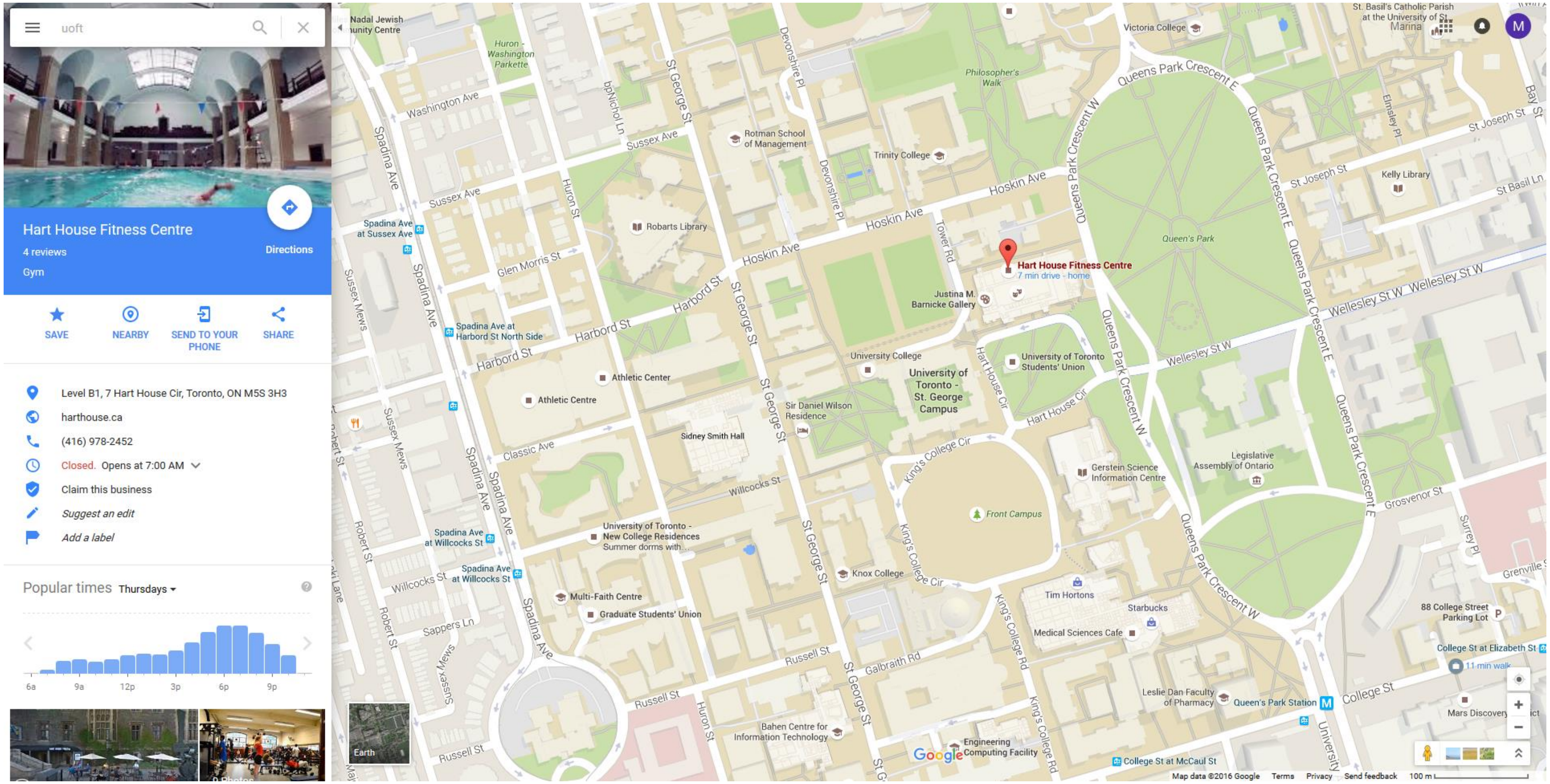
The world of data

- We aggressively acquire and keep data forever
- We feel real freedom when all data is available
- Implications for our live are enormous

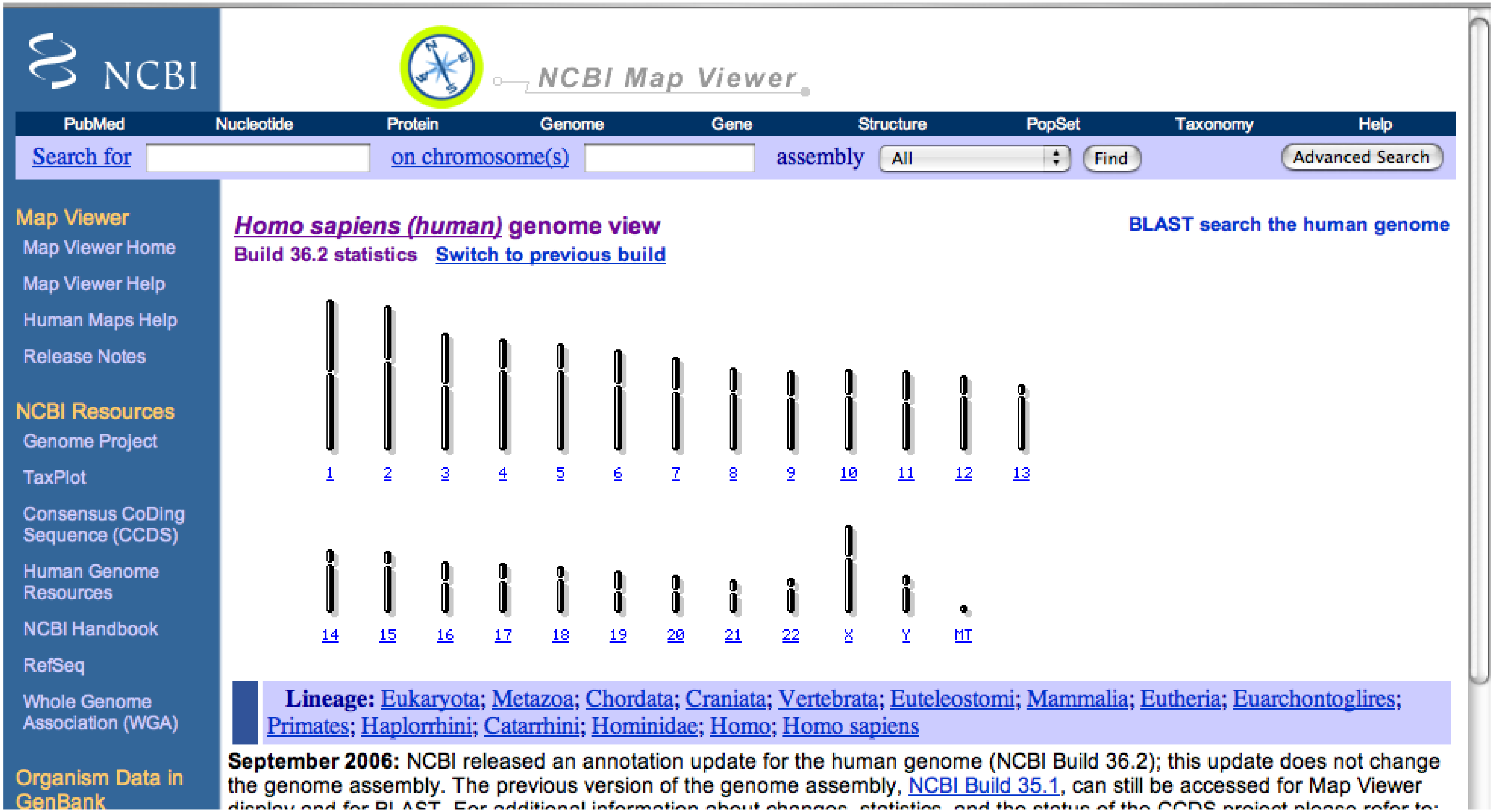
Ubiquitous databases



Ubiquitous databases



Ubiquitous databases



Data science

- Empirical Science – collect and systematize facts
- Theoretical Science – formulate theories and empirically test them
- Computational Science – run automatic proofs, simulations
- **e-Science** (Data Science) – collect data without clear goal - and test theories, find patterns in the data itself

SLOAN DIGITAL SKY SURVEY

What’s a database?

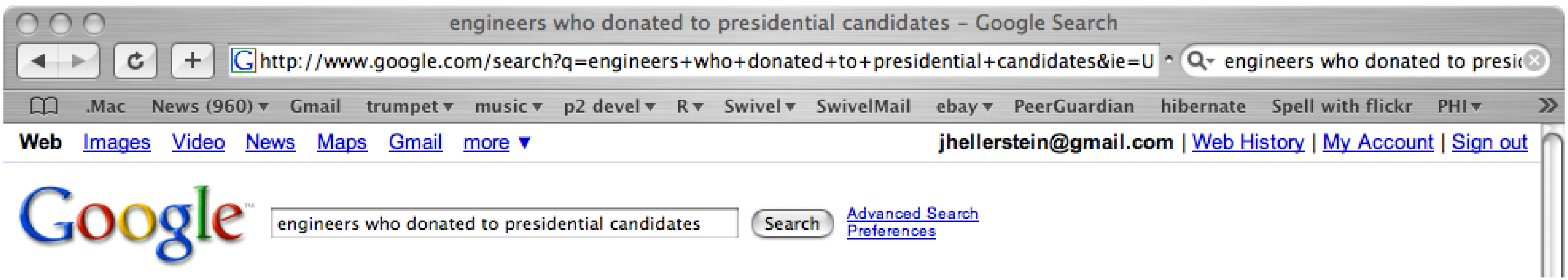
Database (DB): a collection of information that exists over a long period of time.

Is the WWW a database?

- Crawler indexes pages on the web and we can search for pages by keyword
- Source data is mostly “prose”: *unstructured* and untyped
- Public interface is *search only*:
  - can’t modify the data
  - can’t get summaries, complex combinations of data
- Few guarantees provided for freshness of data, consistency across data items, fault tolerance, …

"Search" vs. Query

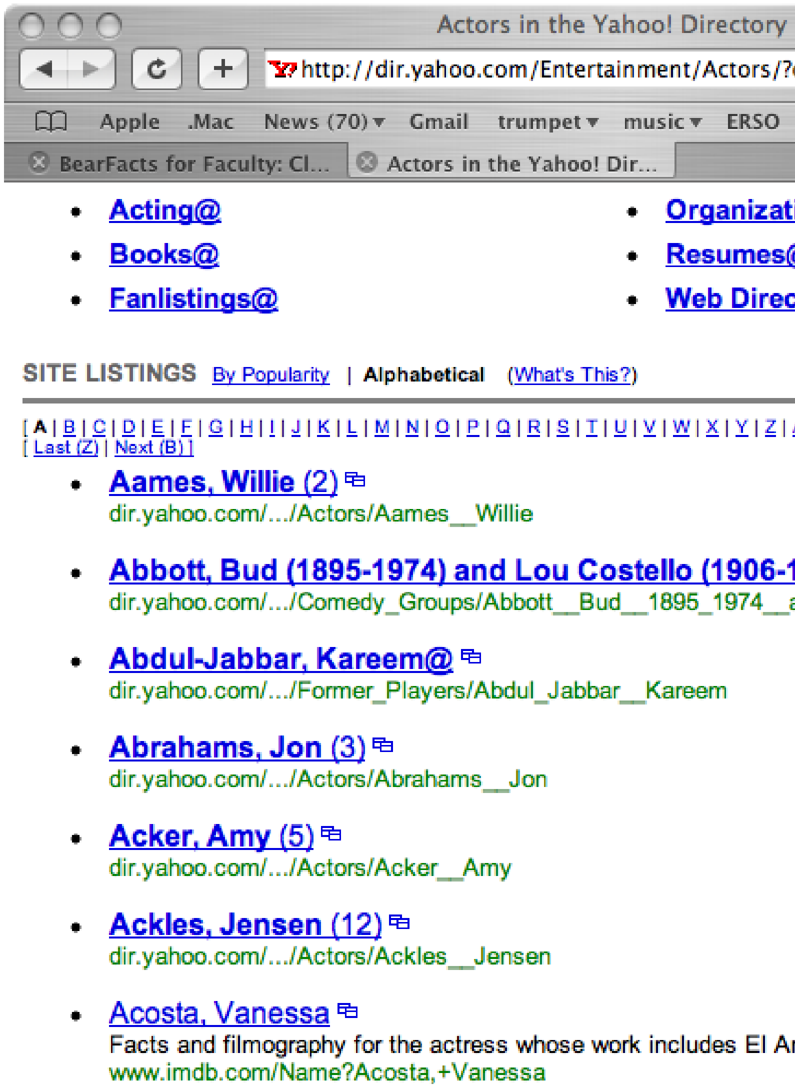
- Try *actors who donated to presidential candidates* in your favorite search engine.
- Now try *engineers who donated to presidential candidates*



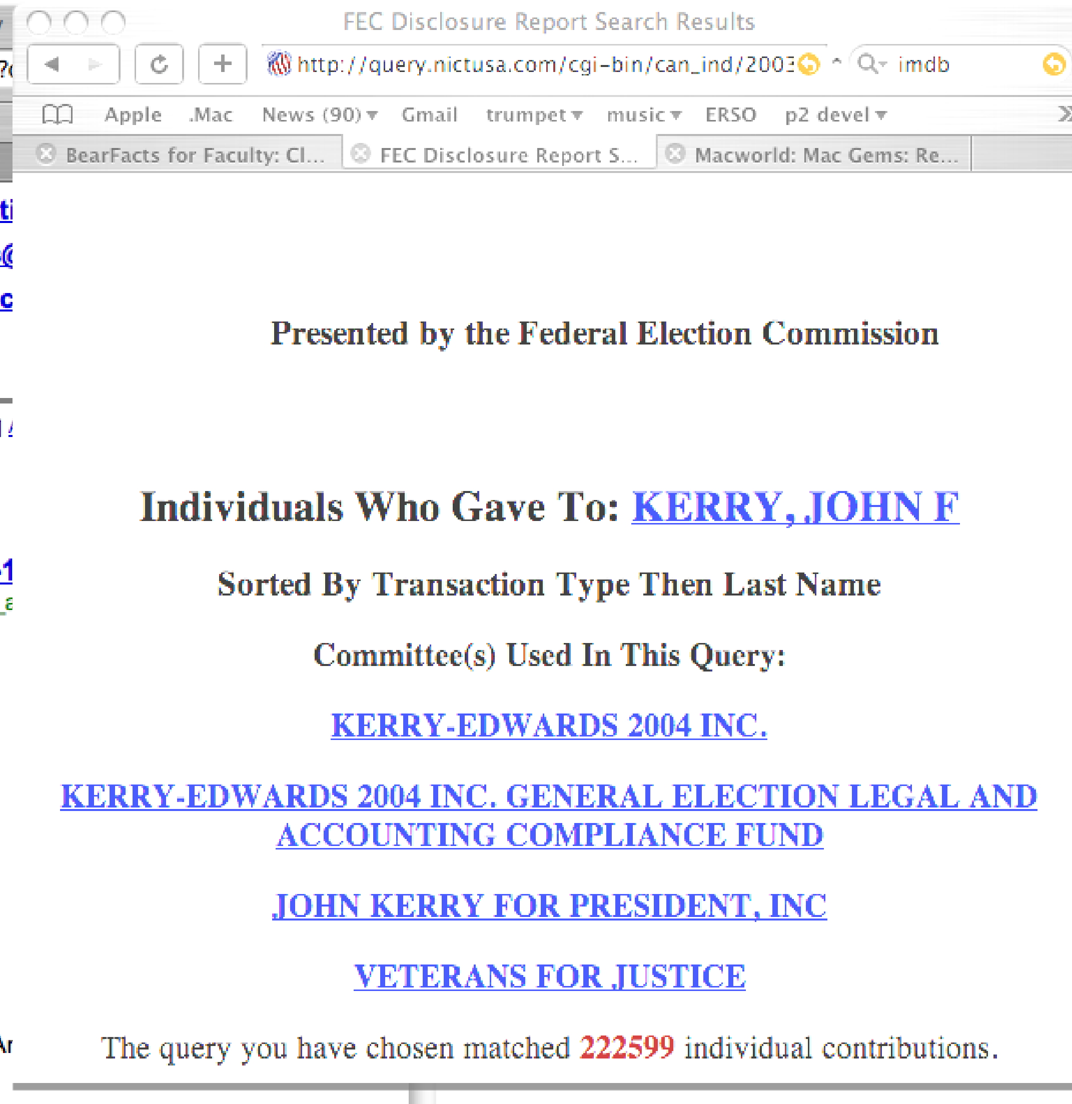
If it isn't "structured", it can't be searched!

A “Database Query” Approach

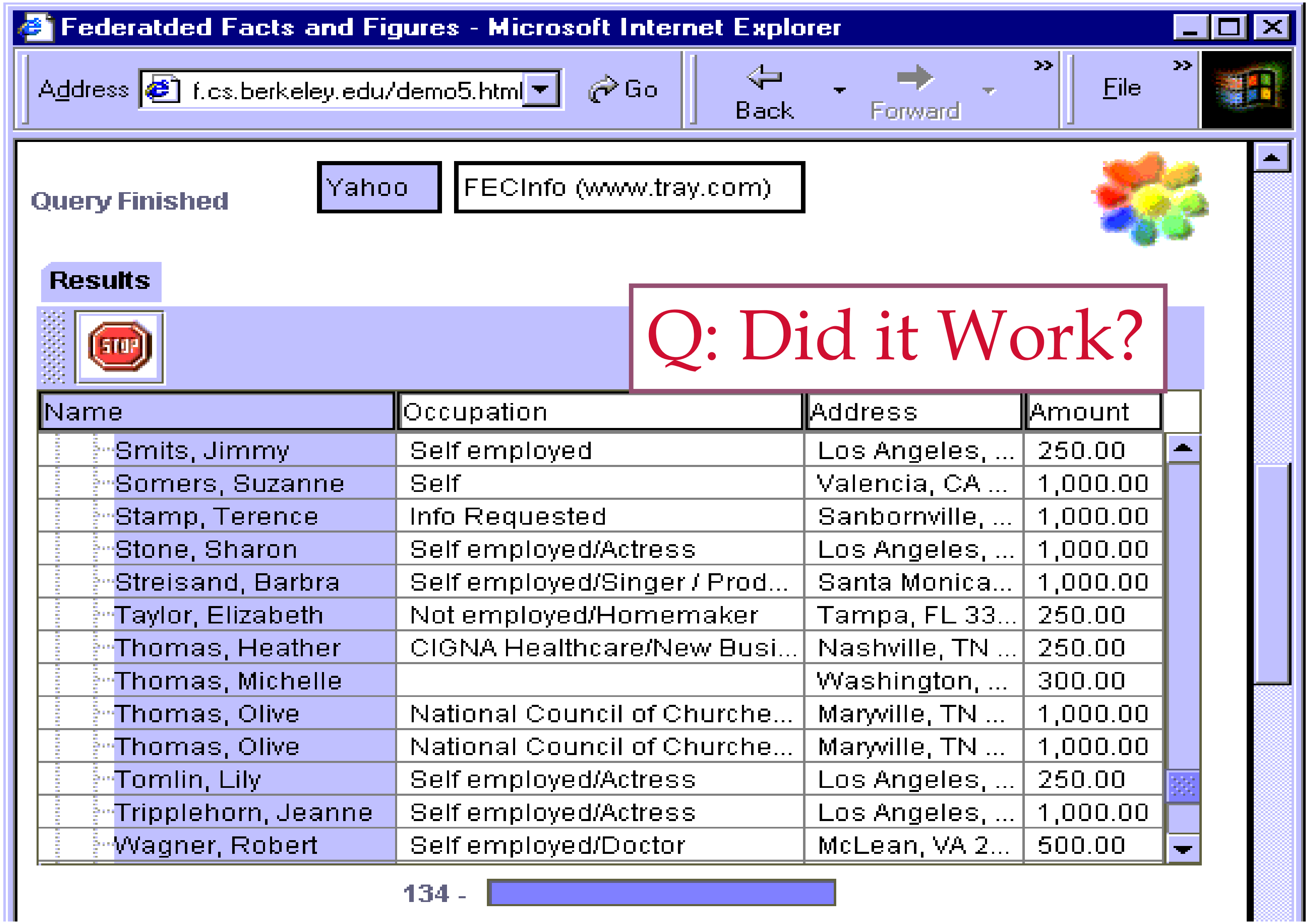
Actors dataset



Donors dataset



“Yahoo Actors” JOIN “FECInfo”



Name	Occupation	Address	Amount
Smits, Jimmy	Self employed	Los Angeles, ...	250.00
Somers, Suzanne	Self	Valencia, CA ...	1,000.00
Stamp, Terence	Info Requested	Sanbornville, ...	1,000.00
Stone, Sharon	Self employed/Actress	Los Angeles, ...	1,000.00
Streisand, Barbra	Self employed/Singer / Prod...	Santa Monica...	1,000.00
Taylor, Elizabeth	Not employed/Homemaker	Tampa, FL 33...	250.00
Thomas, Heather	CIGNA Healthcare/New Busi...	Nashville, TN ...	250.00
Thomas, Michelle		Washington, ...	300.00
Thomas, Olive	National Council of Churche...	Maryville, TN ...	1,000.00
Thomas, Olive	National Council of Churche...	Maryville, TN ...	1,000.00
Tomlin, Lily	Self employed/Actress	Los Angeles, ...	250.00
Tripplehorn, Jeanne	Self employed/Actress	Los Angeles, ...	1,000.00
Wagner, Robert	Self employed/Doctor	McLean, VA 2...	500.00

(From Telegraph research group @Berkeley)

Is a File System a Database?

Thought Experiment 1:

- You and your project partner are editing the same file.
- You both save it at the same time.
- Whose changes survive?

A) Yours B) Partner's C) Both D) Neither E) ???

Thought Experiment 2:

- You're updating a file.
- The power goes out.
- Which changes survive?

A) All B) None C) All Since Last Save D) ???

Is a File System a Database?

- Thought Experiment 1:

file.

A) Y

E) ???

Q: How do you write programs over a subsystem when it promises you only “???” ?

•

–Which changes survive?

A) All B) None C) All Since Last Save D) ???

To have a real database we need to solve problems of:

- **Scale**: data exceeds main memory, specialized (quite complex) EM algorithms, efficiently implemented
- **Sharing**: using the same data by multiple user programs simultaneously (concurrently)
- **Fault-tolerance**: avoiding data loss
- **Consistency**: clean consistent snapshots of data, reinforcing data constraints

Our dream system:

1. Allows to create new databases and specify their schema (logical structure of the data) in a simple language
2. Enables data query and modification, using a simple language
3. Supports intelligent storage of very large amounts of data.
   a. Enforcing constraints (to not allow the insertion of two different employees with the same SIN).
   b. Efficient access to the data for queries and modifications (Indexes).
4. Controls access to data from many users at once (concurrency), without allowing "bad" interactions that can corrupt the consistency.
5. Recovers from software failures and crashes.

Such system exists:

Database Management System (DBMS) - complex *software* for storing and managing databases.

So what is a database?

A *database* is a collection of data managed by a *DBMS*.

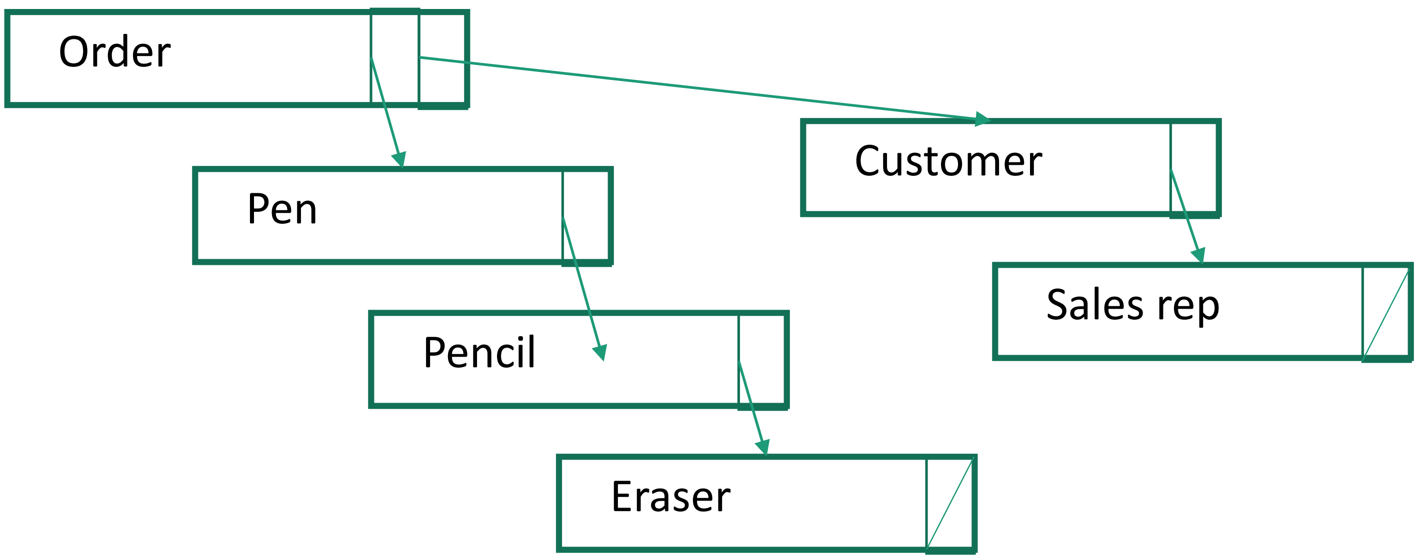
Evolution of Database Management Systems

Early database management systems: files

- First commercial database systems evolved from file systems.
  - File systems allow storage of big amounts of data
  - They do **not** guarantee **data safety**
  - They do **not** resolve an issue of modifying the same file **concurrently**

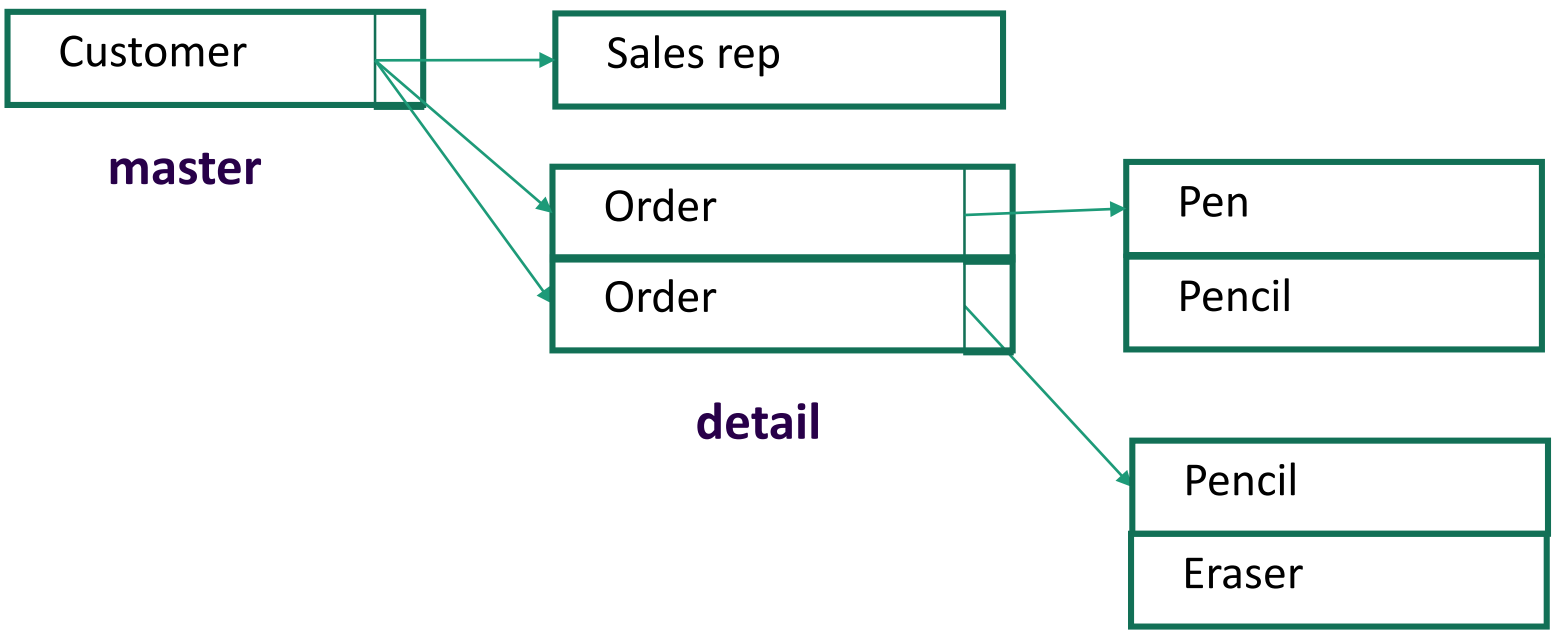
- **No query language** for the data in files.
  - Need to write programs for extracting even the most elementary information from a set of files.

History: network databases (1969)



- Insertions, updates, and deletions are complex and inefficient
- Lack of Data Independence: a change in structure demands a change in the application
- Unanticipated queries cannot be performed efficiently

History: hierarchical databases (1960-s - IBM IMS)



- Data is repetitively stored in many different files.
- Slow search – scan entire model from top to bottom
- One-to-many relationships only (trees)

History: relational databases (1992)

God made the integers;

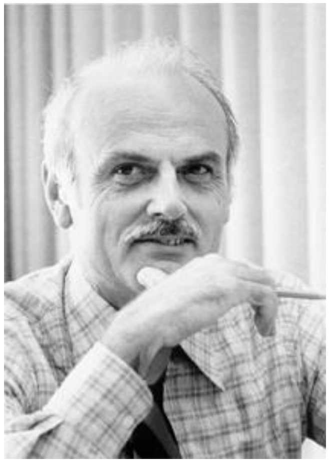
all else is the work of man.

L. Kronecker, 19-th century mathematician

Codd made relations;

all else is the work of man.

R. Ramakrishnan



Edgar Codd
(1923-2003)

Benefits of relational model

Think in terms of tables, not bits on disk.

"Activities of users at terminals ***should remain unaffected when the internal representation of data is changed***."

- Pre-relational: if your data changed, your application broke
- Early RDBMSs were buggy and slow, but required only 5% of the application code

Relational databases: key idea

Programs that manipulate tabular data exhibit an *algebraic structure* allowing reasoning and manipulation independently of physical data representation

Can apply relational algebra!

Ted Codd's vision

- A database system should present the user with a view of data organized as tables (also called *relations*).

- Behind the scene there could be a complex data structure that allows rapid response to a variety of queries. But the user would not be concerned with the storage structure.

- Queries could be expressed in a very high-level language, which greatly increases the efficiency of database programmers.

Example: RDBMS vs Files

- Suppose we have stored in a file called *Employees* records having the fields

  (emp_code, name, dept_code)

- and in another file called *Departments* records having the fields:

  (dept_code, dept_name)

Suppose now that given an employee, for instance with name "Smith", we want to find out what department is he working for.

Files: solution

In the absence of DBMS we have to *write a program* which will:

1. open the file Employees
2. declare a variable of the same type as the records stored in the file
3. scan the file:

    while the end of the file is not yet encountered,

        assign the current record to above variable.

        if the value of the name field is "Smith" then remember the value of the dept_code field. Suppose it is "100"

4. search in a similar way for a record with "100" for the dept_code in the Department file.
5. print the dept_name when successfully finding the dept_code.

Very painful procedure

Modern RDBMS solution

Compare it to the short and elegant SQL query

```
SELECT     dept_name
FROM       Employees, Department
WHERE      Employees.name="Smith" AND
           Employees.dept_code = Department.dept_code
```

Early applications of DBMS's

- Airline reservation systems
- Banking systems
- Corporate records

Data composed of many small items, and various queries and modifications on them.

Case 1: Airline Reservation Systems

- Here the items include:
  - **Reservations** by a single customer on a single flight, including such information as assigned seat…
  - **Flights information** – the airport they fly from and to, their departure and arrival times…
  - **Ticket information** – prices, requirements, and availability.
- Typical queries ask for:
  - Flights leaving about a certain time from one given city to another, seats available, prices.
- Typical data modifications include:
  - Making a reservation in a flight for a customer, assigning a seat, etc.

Case 1: Airline Reservation Systems

- **Many agents** access parts of the data at any given time.

  DBMS must allow ***concurrent accesses*** and prevent problems such as two agents assigning the same seat simultaneously.
- DBMS should also **protect against loss of records** if the system suddenly fails.

Case 2: Banking Systems

- Data items include:
  - **Customers**, their names, addresses etc.
  - **Accounts**, and their balances
  - **Loans**, and their balances
  - **Connections** between customers and their accounts and loans.
- Typical queries are those for account and loan balances.
- Typical modifications are those representing a *withdrawal from* or *deposit to* an account.

Case 2: Banking Systems

- In banking systems **failures cannot be tolerated**.
  - E.g, once the money has been ejected from an ATM machine, the bank must record the debit, even if the power immediately fails.
  - On the other hand, it is not permissible for the bank to record the debit and then not to deliver the money because the power fails.

The proper way to handle this operation is far from obvious and is one of the significant achievements in DBMS architecture.

Example of a Relational DB

- Relations = Tables. Columns are "headed" by *attribute* names.
- Rows = Tuples

A relation Accounts might be:

accountNo	**balance**	**type**
12345	1000.00	savings
67890	2846.92	checking
…	…	…

Queries Examples

1. What's the balance of account "67890" ?
2. Which are the savings accounts with negative balances?

1

```
SELECT balance
FROM Accounts
WHERE accountNo = 67890;
```

2

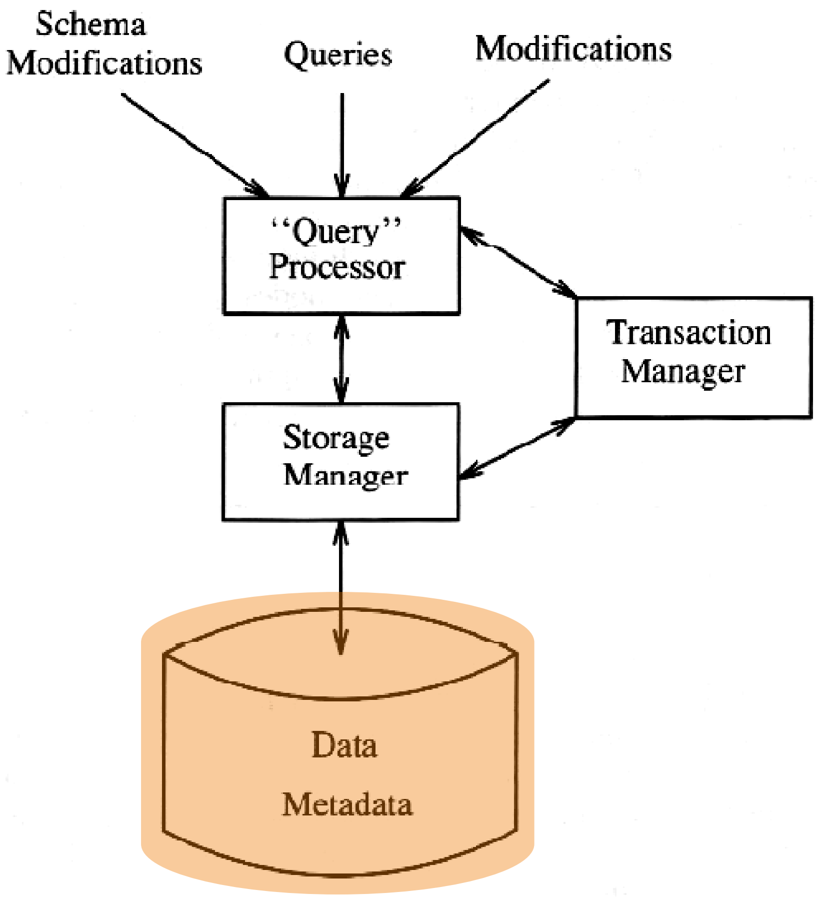
```
SELECT accountNo
FROM Accounts
WHERE type = 'savings' AND balance < 0;
```

Components of a Database Management System

Overview

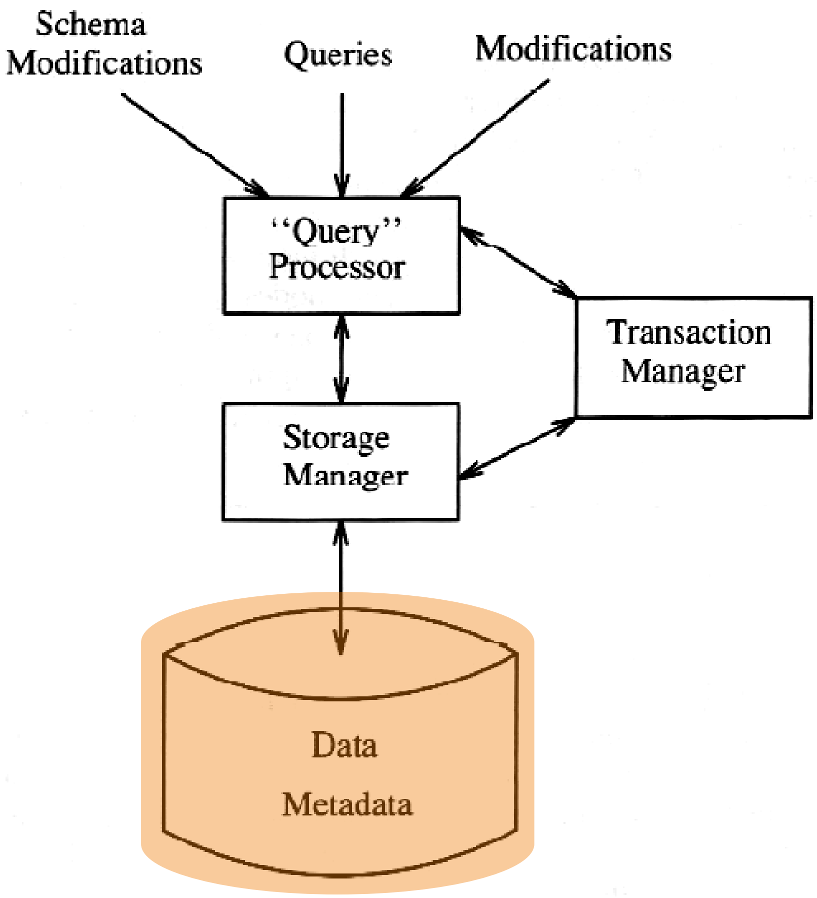
DBMS Architecture

- The "cylindrical" component (representing *persistent storage*) contains not only data, but also metadata, i.e. info about the structure of data.
- If DBMS is relational, metadata includes:
    - names of relations,
    - names of attributes of those relations, and
    - data types for those attributes (e.g., integer or character string).



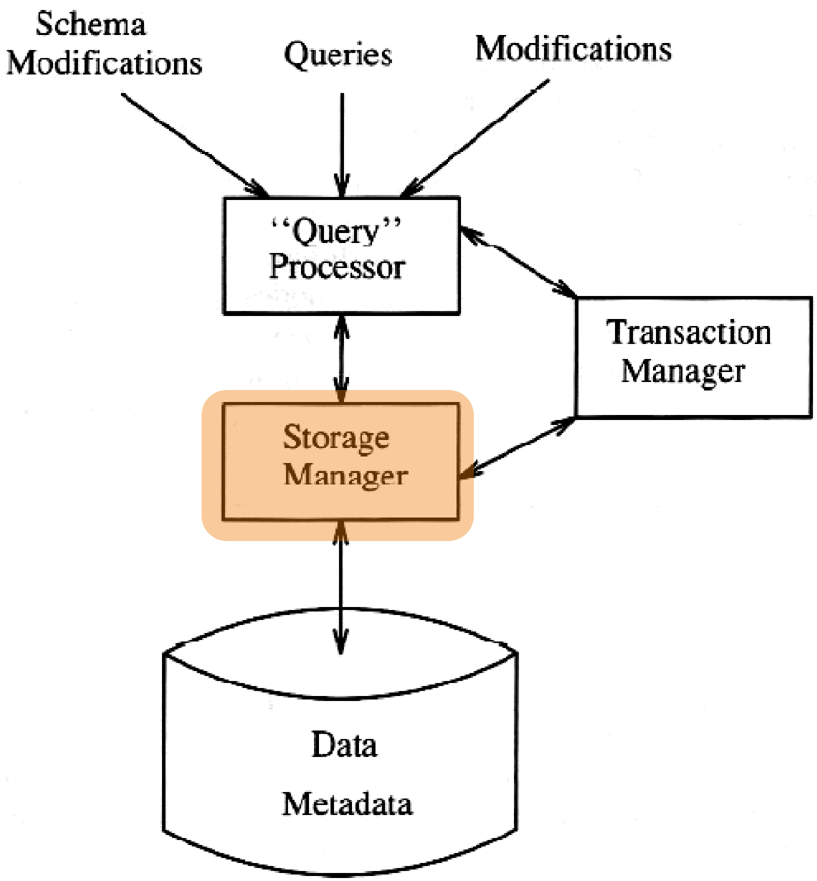
DBMS Architecture

- A database also maintains indexes for the data.
  - Indexes are part of the stored data.
  - Description of which attributes have indexes is part of the metadata.



Storage Manager

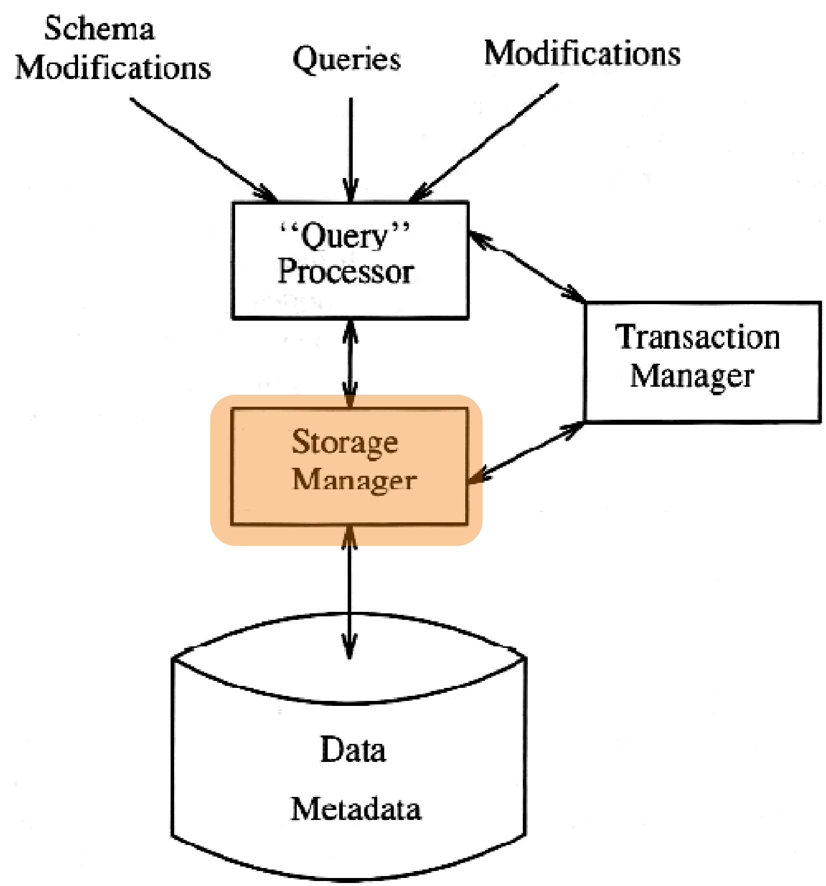
- The job of the **Storage Manager** is to
  - obtain data from the data storage, and
  - return new data to the data storage when updated.
- **Storage Manager** has **two components**:
  - **File Manager** handles on-disk files.
  - **Buffer Manager** handles main memory.



Storage Manager

Storage Manager has **two components**:

- **File Manager** handles files.
  - Keeps track of the location of files
  - Obtains block*(s) of a file on request from the buffer manager.
- **Buffer Manager** handles main memory.



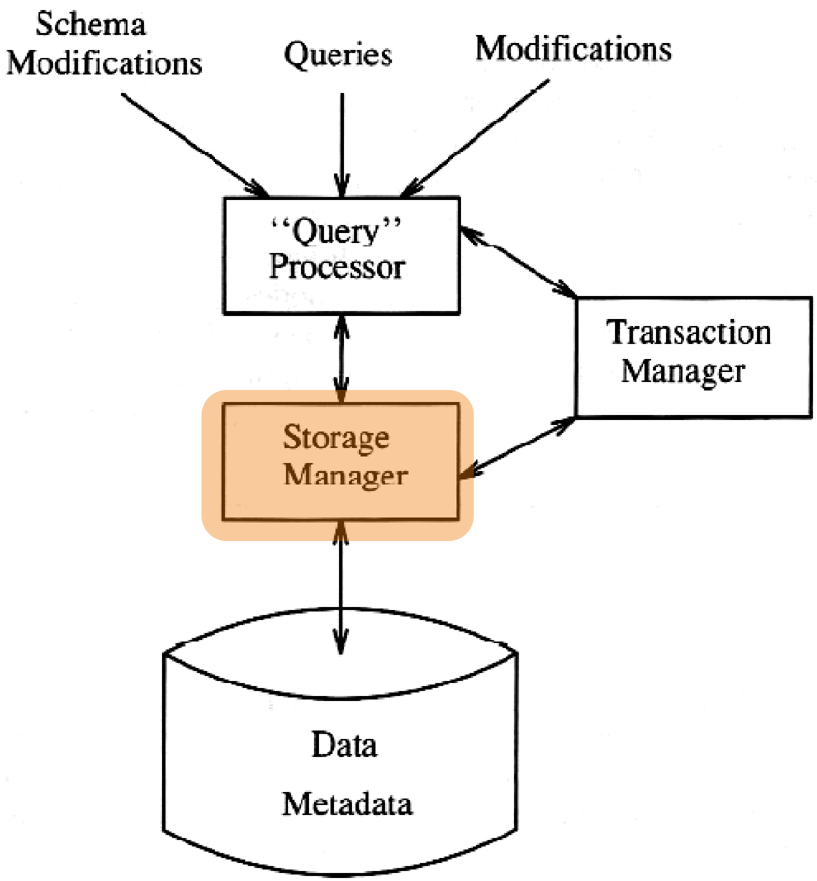
Storage Manager

Storage Manager has **two components**:

- **File Manager** handles files.
  - Keeps track of the location of files
  - Obtains block*(s) of a file on request from the buffer manager.
- **Buffer Manager** handles main memory.

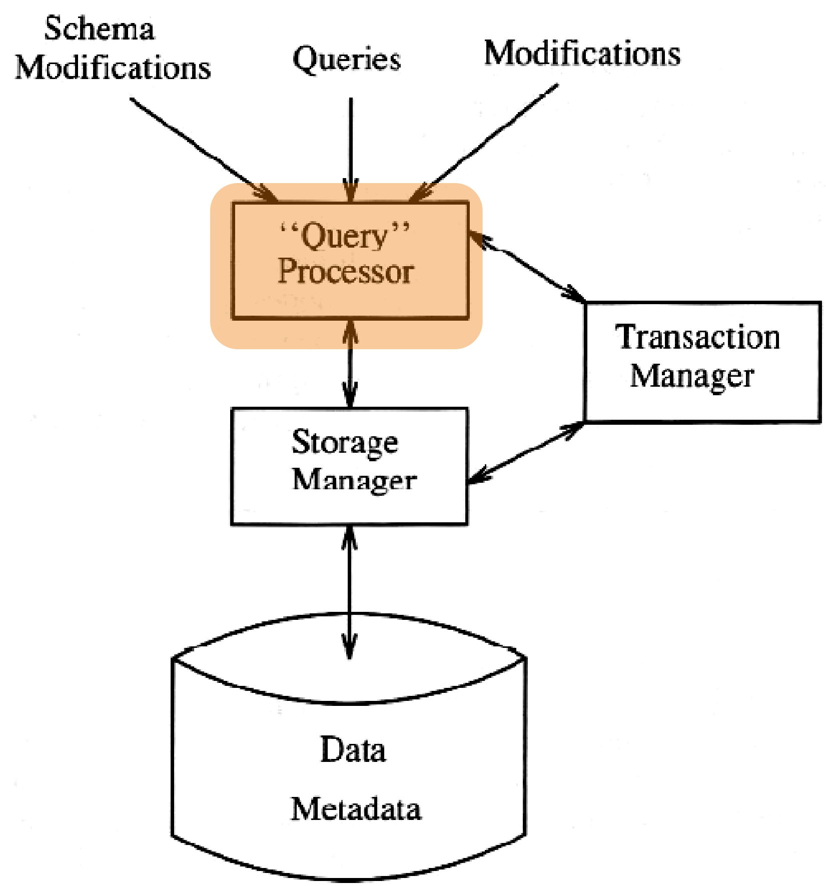
*Block - smallest unit of data that is read/written from/to disk.

1 block = 1 page ≈ 4,000 to 16,000 bytes.



Query Processor

- **Query Processor** handles: queries and modifications to the data.
    - Finds the best way to carry out a requested operation and
    - Issues commands to the storage manager which will carry them out.



Example: Query optimization

A bank has a DB with two tables:

Customers (name, SIN, address),

Accounts (accountNo, balance, SIN)

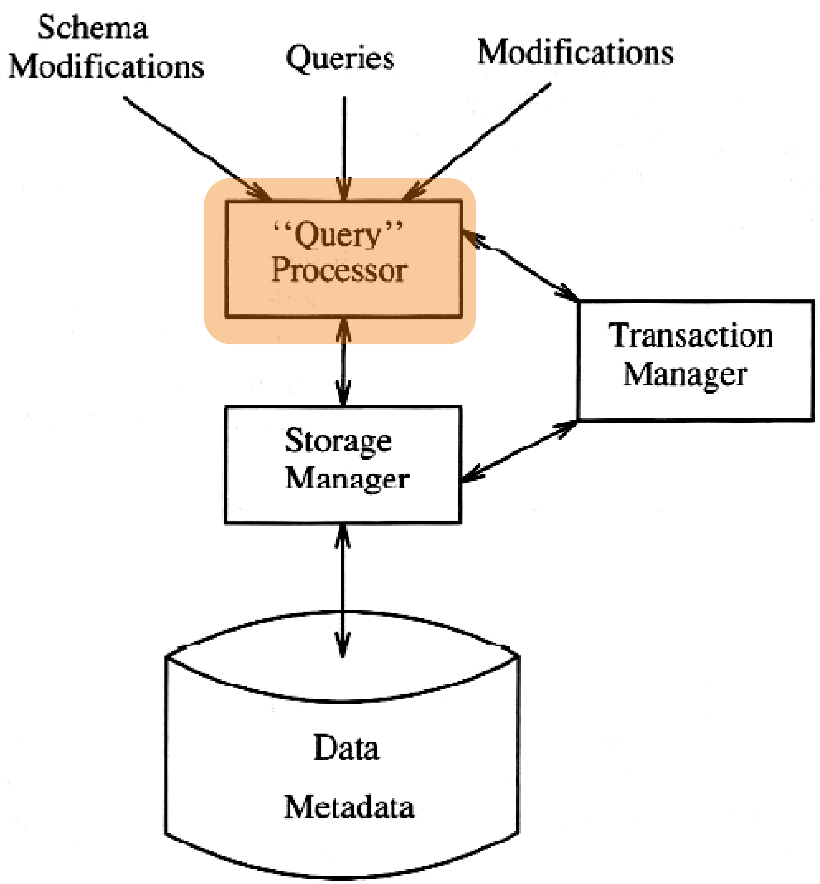
Query: "Find the balances of all accounts of which Sally is the owner."

SQL:

SELECT Accounts.balance

FROM Customers, Accounts

WHERE Customers.SIN = Accounts.SIN
AND Customers.name = 'Sally';



Example: Query optimization

SELECT Accounts.balance

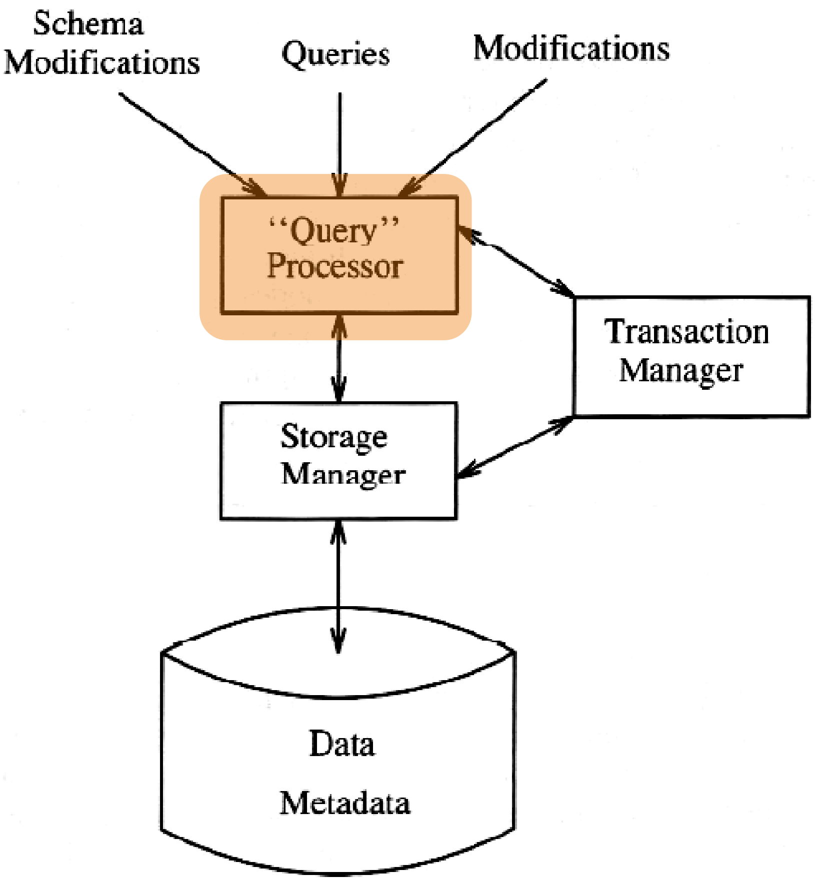
FROM Customers, Accounts

WHERE Customers.SIN = Accounts.SIN
AND Customers.name = 'Sally';

This query - if executed naively:

- Pairs tuples of tables specified in the **FROM**-clause into a new table **R**.
- Chooses from **R** the tuples satisfying the condition in the **WHERE** clause.
- Produces as answer only the values of attributes in **SELECT**-clause.

The performance would be terrible, because of the usually enormous (quadratic) size of all pairs of tuples.



Example: Query optimization

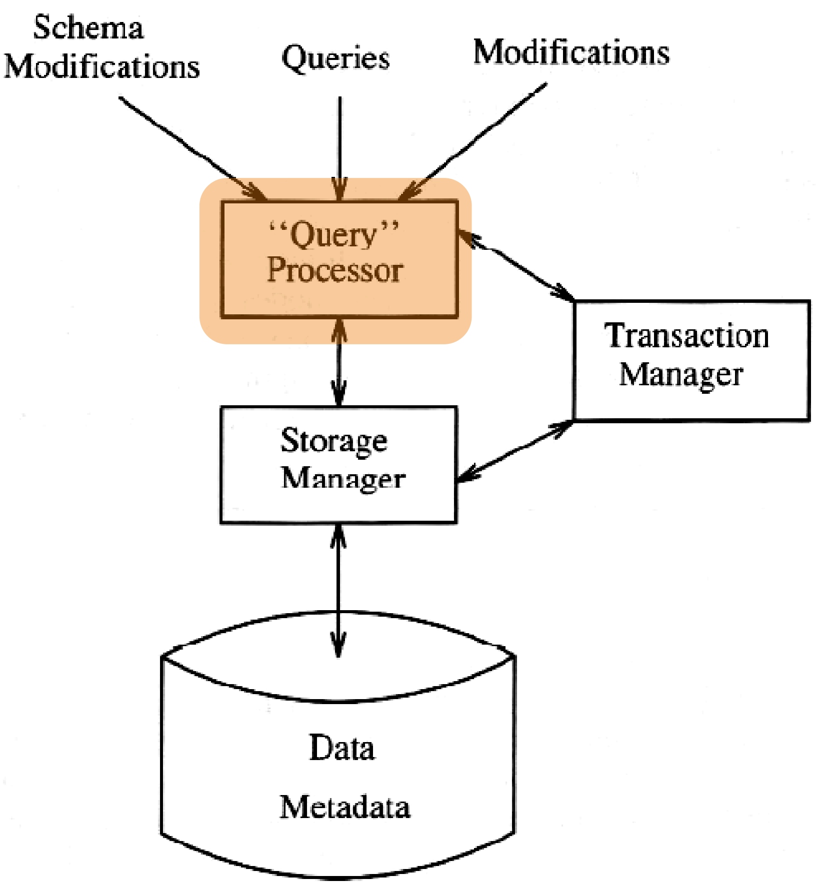
SELECT Accounts.balance

FROM Customers, Accounts

WHERE Customers.SIN = Accounts.SIN
AND Customers.name = 'Sally';

Query processor will cleverly create a plan which inexpensively:

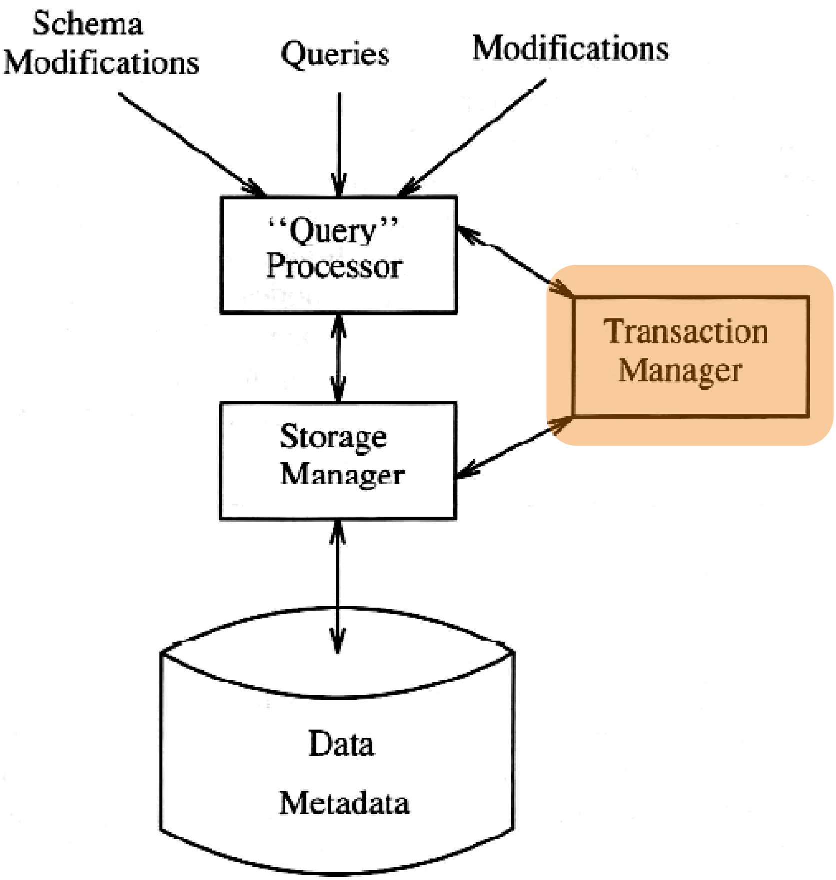
- Retrieves the tuple for "Sally" and gets the SIN number
- Retrieves the account tuples for this SIN number



Transaction manager

Transaction Manager assures that:

- several queries running simultaneously do not interfere with each other and that,
- the system will not end up with corrupted data even if there is a power failure.



Transaction manager

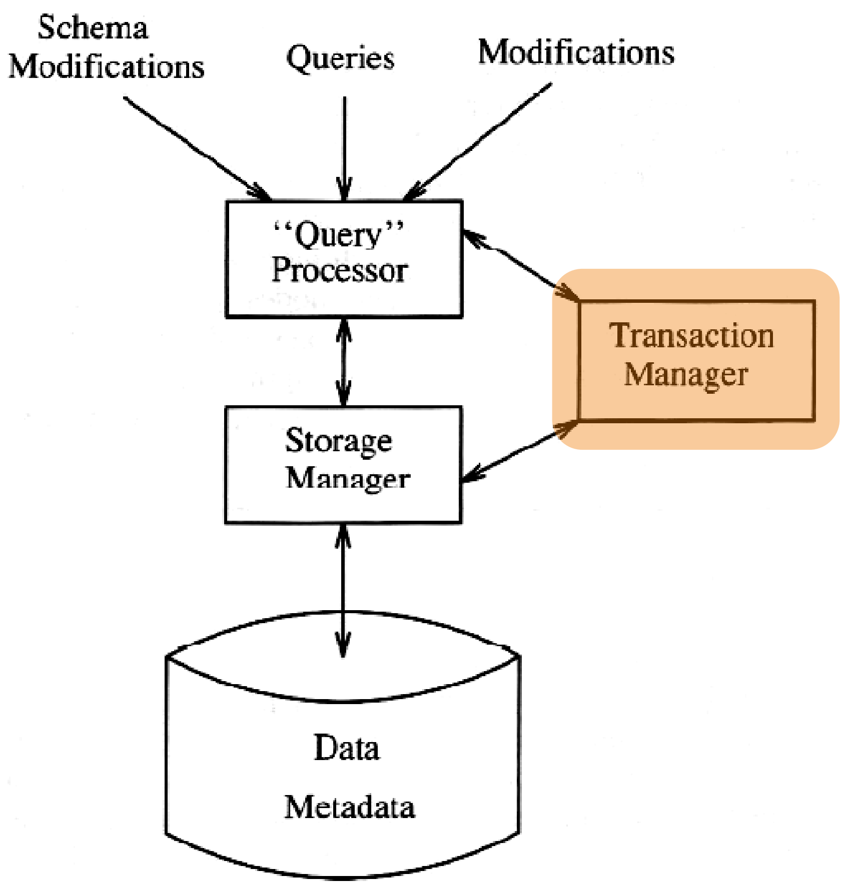
Transaction Manager interacts with:

- **Query Manager**

  Because it may need to delay certain query operations to avoid conflicts.

- **Storage Manager**

  Because schemes for protecting data involve storing a **LOG** of changes to the data.



DBMS is a very complex system

Good news: it has been already built for you to use

Modern RDBMS's guarantee:

- Efficient algorithms for out-of-memory inputs
- Enforcing consistency of data
- Data safety
- Multi-user concurrency
- Convenient interface – level of abstraction above physical data storage: declarative language SQL

Database studies

- Design of databases (data modeling).
  - How to structure information?
  - How to connect data items?
  - What constraints should the data satisfy?
- Database programming.
  - How to query and modify the database?
  - How is database programming combined with conventional programming?
- Database system implementation.
  - How does one build a DBMS, including such matters as query processing, transaction processing and organizing storage for efficient access?

Database studies

- Design of databases (data modeling).
  - How to structure information?
  - How to connect data items?
  - What constraints should the data satisfy?

- Database programming.
  - How to query and modify the database?
  - How is database programming combined with conventional programming?

That is in CSC443

- Database system implementation.
  - How does one build a DBMS, including such matters as query processing, transaction processing and organizing storage for efficient access?

In this course we explore database world from the point of view of:

Designer

Developer

User

Main topics

- Database design and data modeling
- Data storage and manipulation through DBMS
- Queries about data in relational model
- Alternative data models and their applications
- Embedding databases into conventional programs

Both theory and practice!

DBMS`s used for this course

- PostgreSQL: advanced open source database with enterprise-class features comparable to Oracle and DB2
- SQLite: self-contained, serverless, zero-configuration, transactional SQL database engine, for small data collections
- MongoDB: free and open-source cross-platform document-oriented database

Course mechanics

- 10 Homework assignments: 10%
- 3 major project-oriented Assignments: 45%
- Midterm: 10%
- Final exam: 35%

Textbook: A First Course in Database Systems *by Jeffrey D. Ullman, and Jennifer D. Widom,* Pearson, 3-rd edition, SBN-10: 013600637X, 2008.

Why take this class?

A. Database systems are at the core of CS
B. They are incredibly important to society
C. The topic is intellectually rich
D. It isn't that much work
E. Looks good on your resume
F. Be a data ninja

Socrative.com Room: **OLKOQEWE**

Why take this class?

A. Database systems are the core of CS

- Shift from computation to information
  - True in corporate computing for years
  - Web made this clear for “the rest of us” by the end of 90’s
  - Increasingly true in scientific computing
- Need for DB technology has exploded
  - Corporate: retail swipe/clickstreams, “customer relationship mgmt”, “supply chain mgmt”, “data warehouses”, etc.
  - Web: not just “documents”. Search engines, maps, e-commerce, blogs, wikis, social networks.
  - Scientific: digital libraries, genomics, satellite imagery, physical sensors, simulation data
  - Personal: Music, photo, & video libraries. Email archives. File contents (“desktop search”).

Why take this class?

B. DBs are incredibly important to society

- Policy-makers should understand technological possibilities
- Informed Technologists needed in public discourse
- Everyone should be provided with access to data

"*Knowledge is power*." -- Sir Francis Bacon

"*With great power comes great responsibility*." -- Spiderman's Uncle Ben

Why take this class?

C. The topic is intellectually rich.

- Sophisticated algorithms for massive data
- Complex system architecture and implementation
- Resource management and scheduling of concurrent transactions
- Query language design, semantics and optimization
- Data modeling
- Data analytics

Why take this class?

~~**D. It isn't that much work.**~~

- Bad news: It is a fair bit of work.
- Good news: it is a lot of fun (at least in the eye of the instructor)

Why take this class?

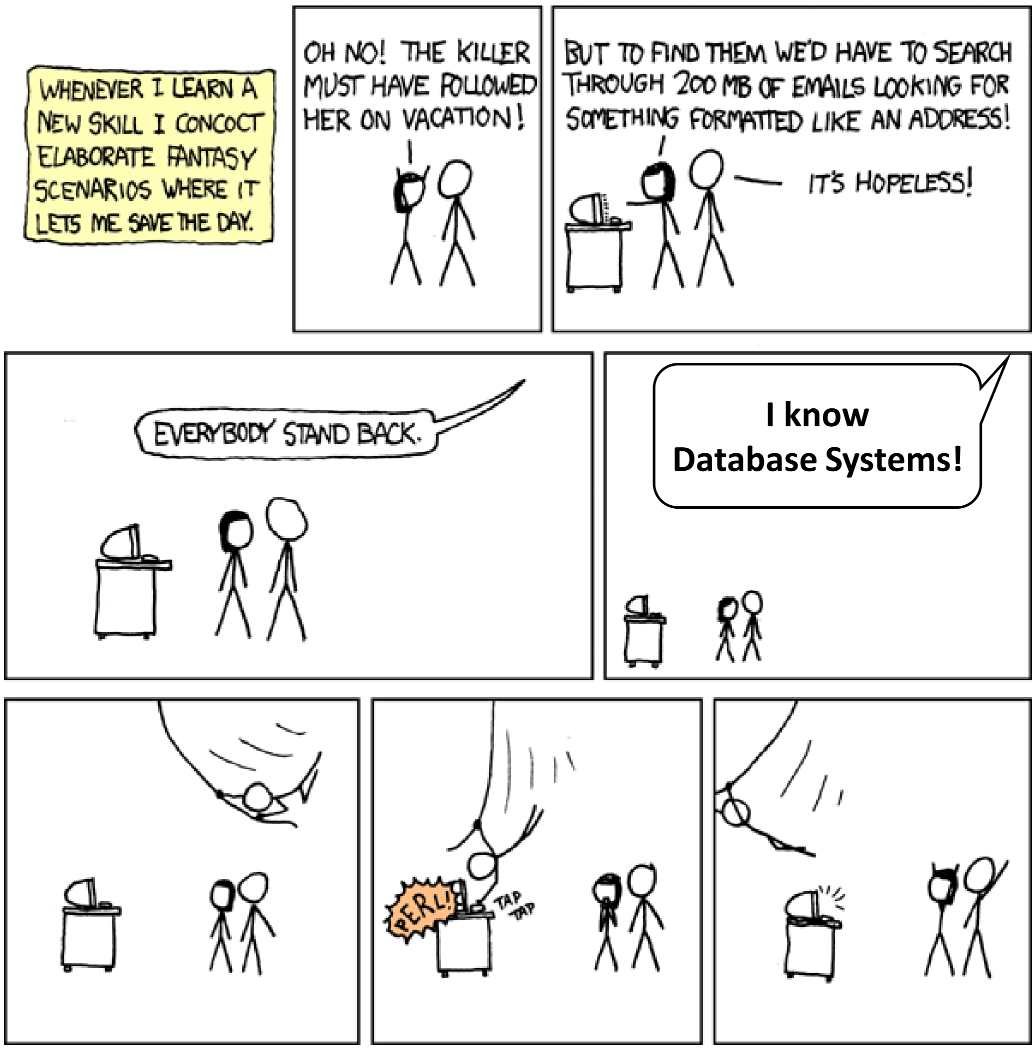
E. Looks good on my resume.

Yes, but why?

- Data Management is simultaneously the most boring and most interesting technology around!
- Database systems are “merely” a means to an end.
- We want cool applications.
- …how long to prototype/build your new application?
- …how long to add features?
- …what happens when the power goes out, disk crashes, etc? (cool applications don’t lose user data)

Why take this class?

F. Be a data ninja.



I can handle data quickly, flexibly and powerfully – just like a ninja

xkcd.com/208