CS 236 Database Management

Introductions …

Prof. Mariam Salloum

Office: Bourns Hall A (Room 159B)

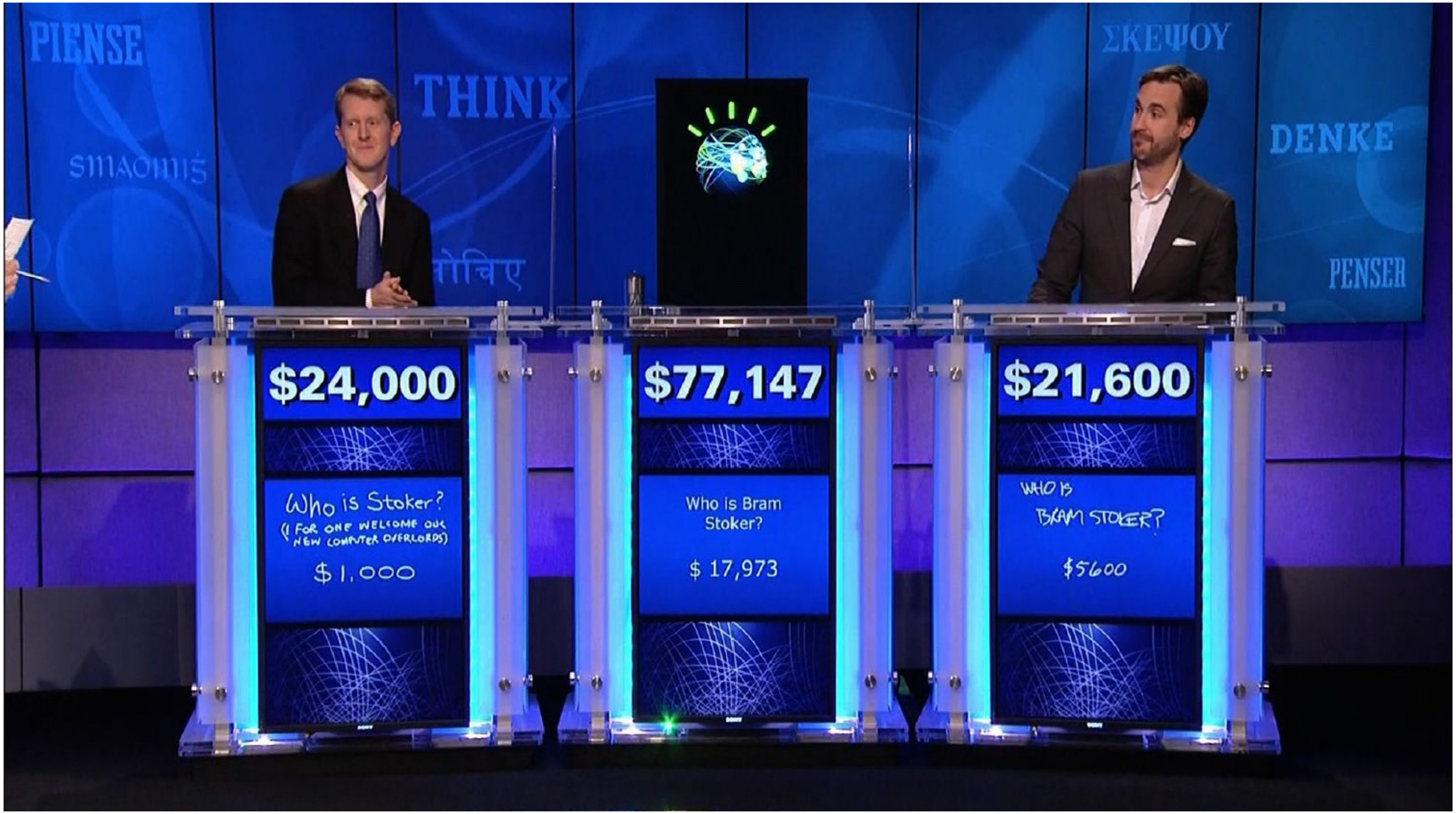
Email: msalloum@cs.ucr.edu

- A UCR grad – received my PhD under the guidance of Prof. Tsotras
- Research:
  - **Data Integration :** Given millions of relevant sources/sites to a particular query, how to identify relevant sources, query sources in optimized fashion then perform record linkage and data fusion.
  - **Applied ML: Working on real datasets with research labs.**
    - Building ML models on satellite imagery for NASA JPL to localize flooded regions;
    - Building ML models on satellite imagery to understand farm land terrain / patterns (NASA JPL project);
    - Building ML models to label anomalies in flight time series (NASA Ames project).

What is this course about?

Why take it?

IBM Watson beat humans in Jeopardy

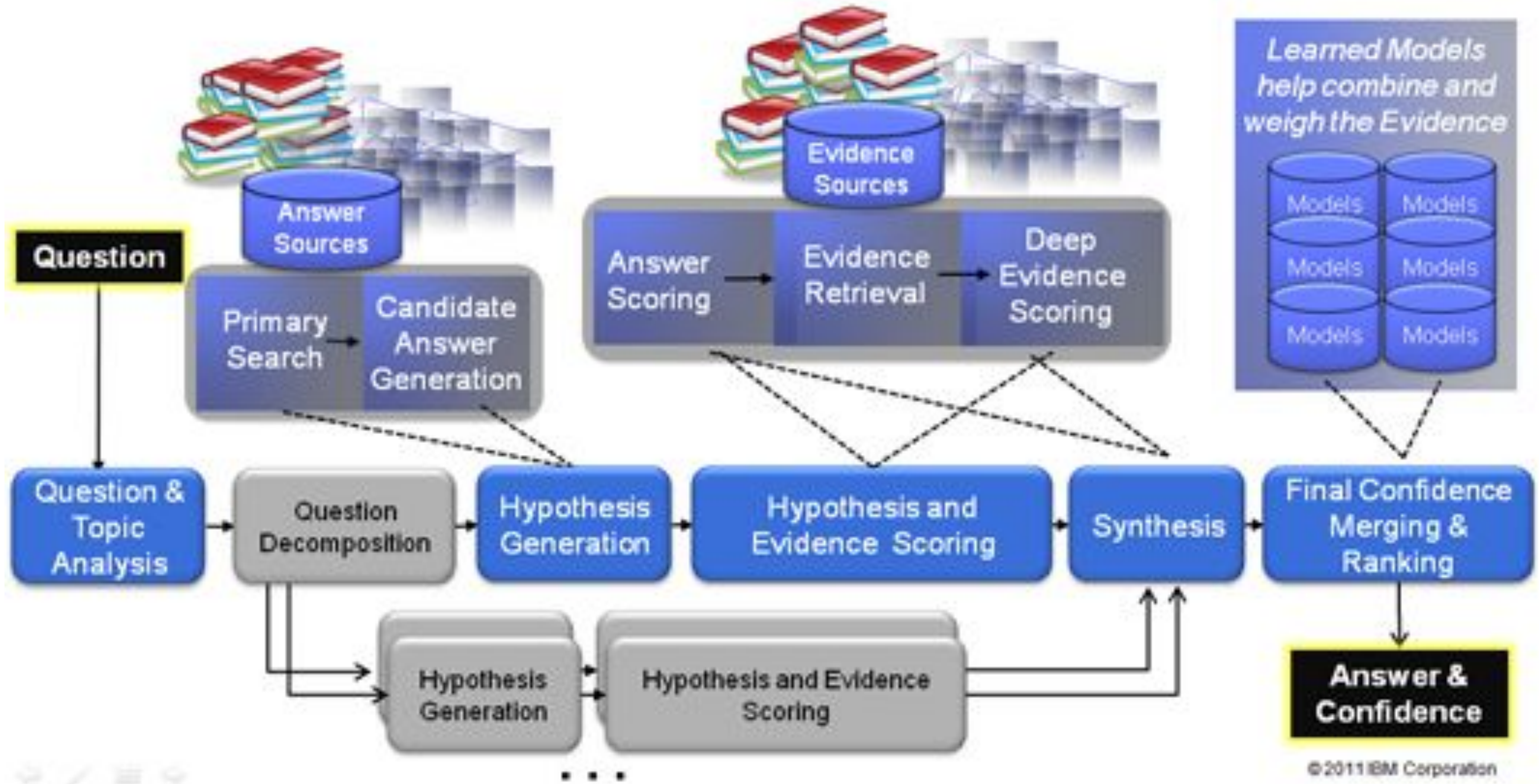


How did it accomplish this??

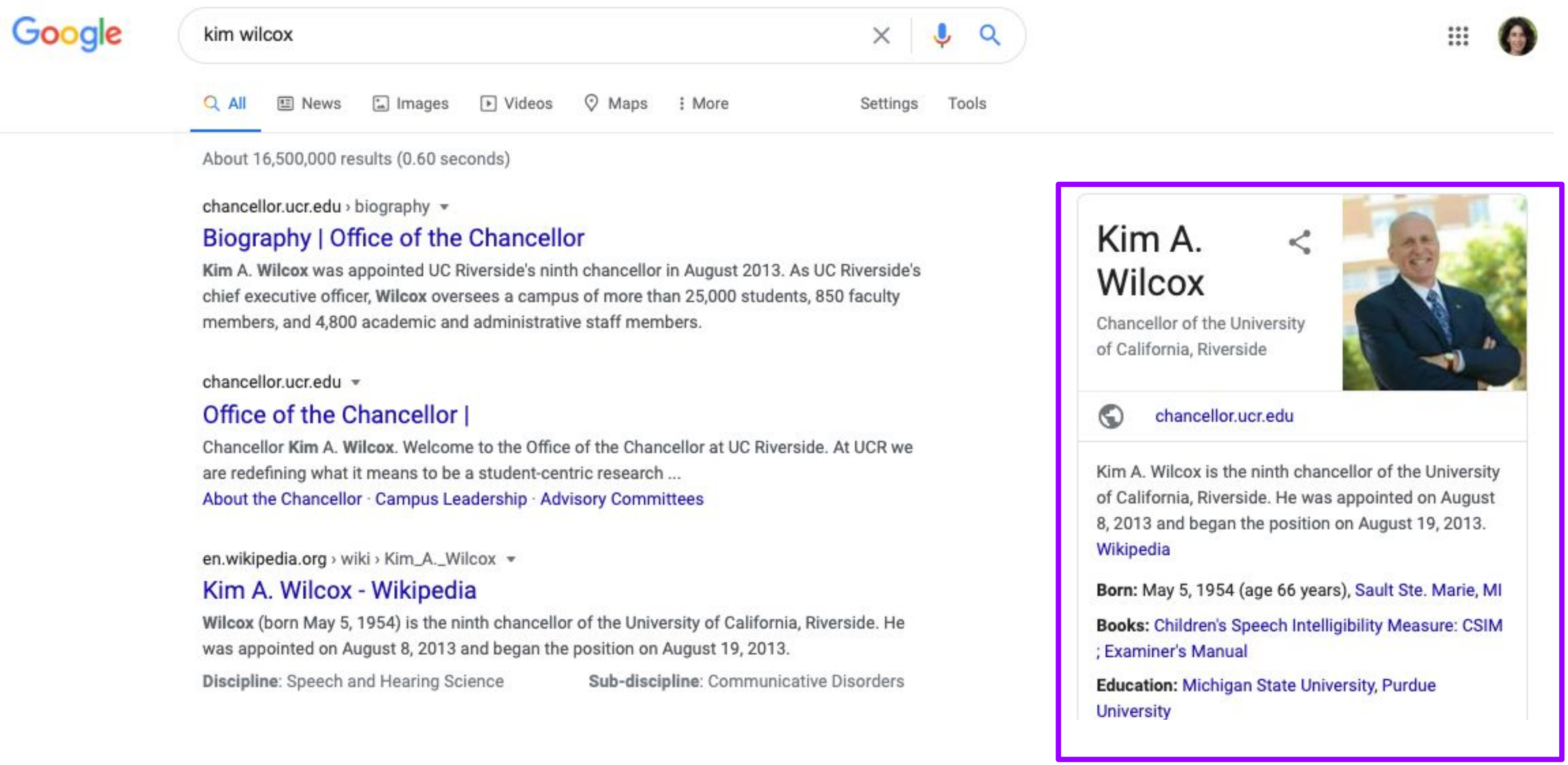
Watson devoured alot of data!!

Read this IBM Article

High Level View of DeepQA Architecture -



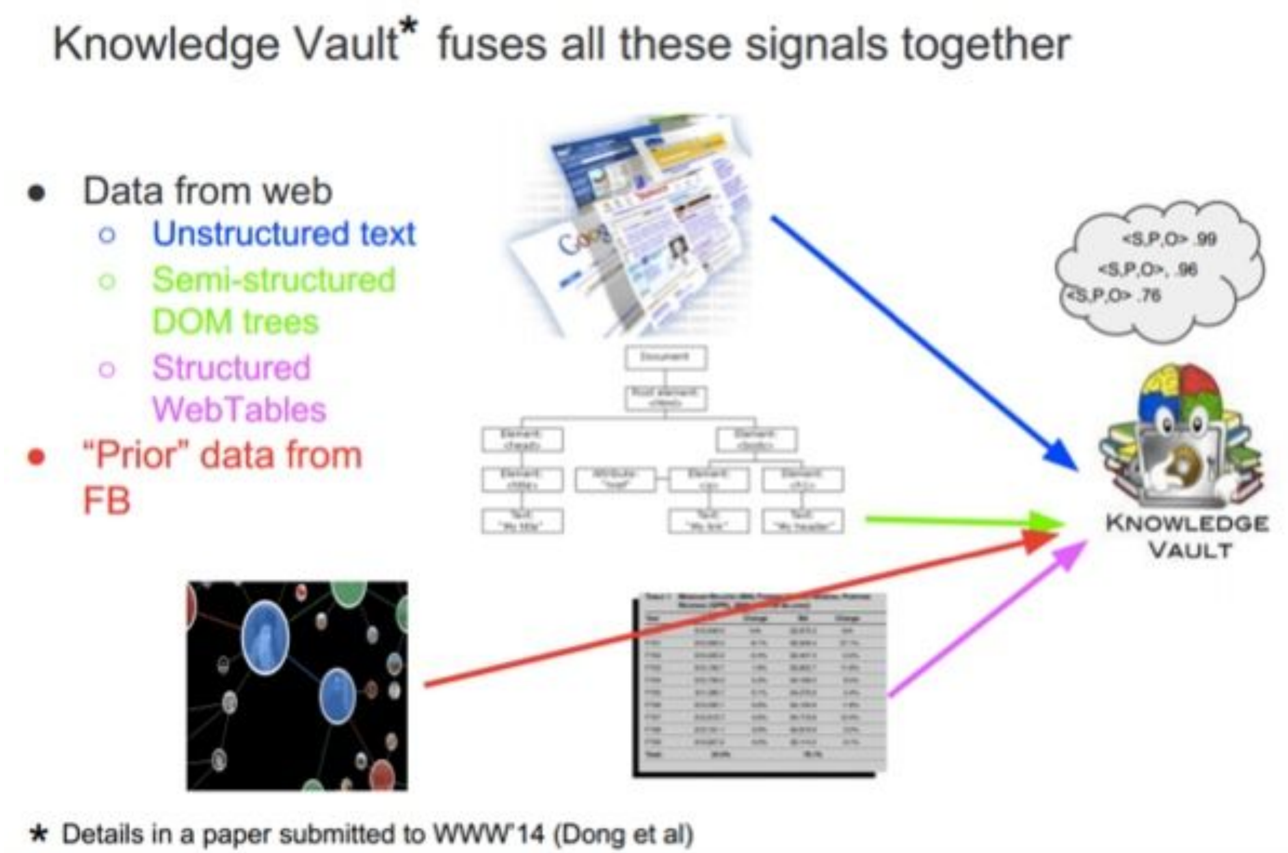
“Structured” data with Google search results



How does Google generate this?

Google also devours LOTS of data!

[Read this Google paper about knowledge Vault]



What this course is about?

Large-scale data management systems are the cornerstone of many digital applications, both modern and traditional.

This course will get you thinking about the fundamentals of database systems:

1. How are large structured datasets stored and organized?
    - Storage and file layout
2. How are “queries” handled?
    - Indexing, sorting, relational operator implementations, and query processing
3. How to make the system faster?
    - Query optimization and parallelism
4. Deeper and more recent issues
    - Data systems for ML, Data integration and cleaning, semi-structured data, ML for RDBMSs

Course Logistics

- Canvas (elearn.ucr.edu) :
  - Will be used to post grades
- Lectures / Labs
  - **Lecture:** T/TH2:00PM - 3:20PM
- My Info:
  - **Email:** msalloum@cs.ucr.edu
  - **Website:** www.cs.ucr.edu/~msalloum
  - **Office Hours:** MW 4-5 and by appointment

Class Policies

Please attend lecture and participate in discussions

Please update your picture on Zoom

Please attend lecture on time

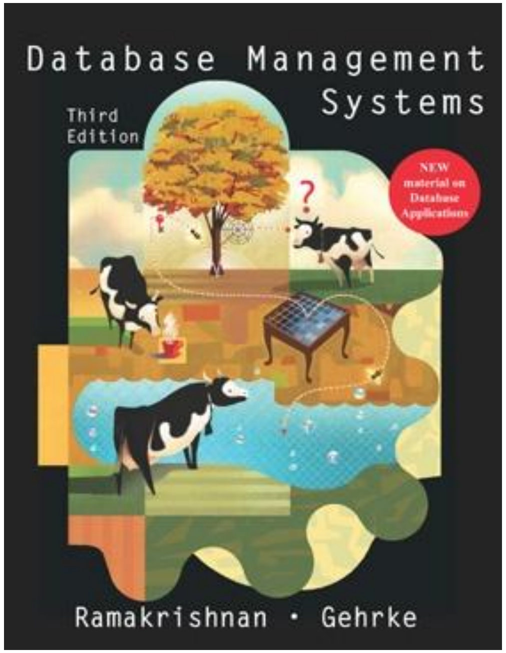
- Lectures will be recorded and recordings provided on demand

Textbook

http://pages.cs.wisc.edu/~dbbook/

Contains many solved exercises:
http://pages.cs.wisc.edu/~dbbook/openAccess/thirdEdition/solutions/ans3ed-oddonly.pdf

3rd Edition



Aka The “Cow Book”

Alternative Textbooks

“Database Systems: The Complete Book” by Garcia-Molina, Widom, and Ullman

(eBook on UCR Library Website - just make sure you are connected to UCR VPN)

Requirements & Grading

Item	Percentage	Notes
Midterm	25%	Midterm during scheduled class time (unless other arrangements are made)
Final Exam	30%	Cumulative final exam
Assignments	30%	Written and programming assignments
Paper Reviews	15%	Reviews of recent research papers in top databases conferences like SIGMOD or VLDB.

Assignments

- We will have both written and programming assignments.
- All written assignments will be turned in via Canvas
- Programming assignments will be turned in via Gradescope.
  - You will receive an invite email later this week.
- No late assignments will be accepted, unless accompanied by a doctor’s note.

Cheating Policy

Students must read and understand UCR policy on academic honesty. Look at:

https://conduct.ucr.edu/policies/academic-integrity-policies-and-procedures

Read the Guidelines & Definitions, and download the *Academic Integrity Student Brochure*

Anyone caught cheating will get a final grade of **F** and may have a letter placed in his/her permanent record. Students are expected to take care that others cannot "cheat off them". For example, if your homework is left on a shared hard drive and someone else hands it in, you are liable and will have your grade adjusted downward.

Topics

- Ch.1 -2: Introduction to Database Design
- Ch.3: The Relational Model
- Ch.4: Relational Algebra
- Ch.5: SQL
- Ch.8 -9: Storage and Indexing
- Ch.10 - 11: Tree-Structured & Hash-Based Indexing
- Ch.12: Overview of Query Evaluation
- Ch.13: External Sorting
- Ch.14: Evaluation of Relational Operators
- Ch.15: A Typical Relational Query Optimizer
- Ch. 16: Transaction Management
- Ch. 19: Normalization
- Ch. 22: Parallel and Distributed Databases
- Ch. 28: Spatial Data structures / Queries /Joins

- Advanced topics (examining papers in recent conferences)

The primary focus will be the relational data model and Relational Database Management Systems (RDBMS)

But isn't NoSQL the new craze??

CS226 Big Data Management goes over such platforms like NoSQL and Hadoop.

But it is critical to understand the fundamentals before diving into other topics. Read this post by Stonebraker and DeWitt (LINK)

Intro to Databases

Can I just use files to store data???

Winter 2020 Schedule

Jane Smith , CS166 , Database Management , TH

John Smith , CS105 , Intro to DS , MW

John Smith , CS166 , Database Management , TH

Kate Aron , CS172 , Information Retrieval, MWF

Student File

Jane Smith , 123 boxwood 91823

John Smith , 451 lemonswirl 91709

Kate Aron , 925 buttonwood 91703

What happens if Kate drops CS 172? Do we lose all information about CS172??

Can I just use files to store data???

Student File

Jane Smith , 123 boxwood 91823

John Smith , 451 lemonswirl 91709

Kate Aron , 925 buttonwood 91703

Winter 2020 Schedule

Jane Smith , CS166 , Database Management , TH

John Smith , CS105 , Intro to DS , MW

John Smith , CS166 , Database Management , TH

Kate Aron   , CS172 , Information Retrieval, MWF

Course File

CS166 , Database management, TH

CS 105 , CS105 , Intro to DS , MW

CS 172 , CS172 , Information Retrieval, MWF

What happens if we want to rename the course ?

Do we only do it in one place, or do we have to do it in many places?

What is an Anomaly?

- Definition
  - Problems that can occur in poorly planned, un-normalized databases where all the data is stored in one table
  - Ex. One big file

- Types of anomalies
  - Insert
  - Delete
  - Update

Insert Anomaly

- An **Insert Anomaly** occurs when certain attributes cannot be inserted into the database without the presence of other attributes.

Course _no	Tutor	Room	Room_size	En_limit
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	H940	400	45

e.g. we have built a new room (e.g. B123) but it has not yet been timetabled for any courses or members of staff.

Delete Anomaly

- A **Delete Anomaly** exists when certain attributes are lost because of the deletion of other attributes.

Course_no	Tutor	Room	Room_size	En_limit
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	H940	400	45

e.g. if we remove the entity, course_no:351 from the above table, the details of room C320 get deleted. Which implies the corresponding course will also get deleted.

Update Anomaly

- An Update Anomaly exists when one or more instances of duplicated data is updated, but not all.

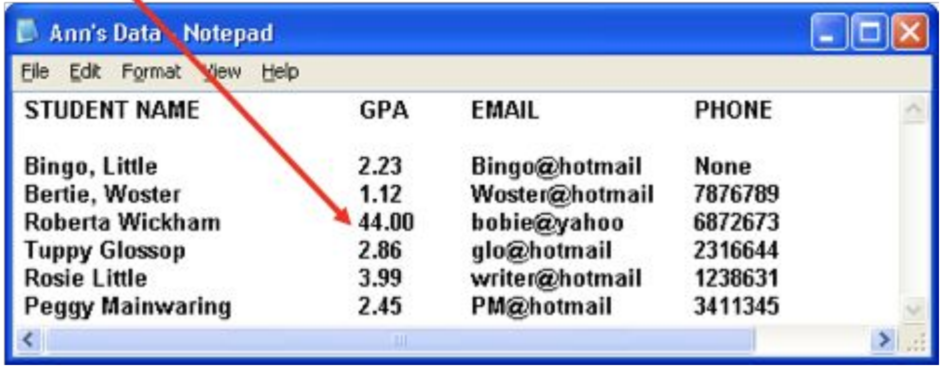
Course_no	Tutor	Room	Room_size	En_limit
353	Smith	A532	45	40
351	Smith	C320	100	60
355	Clark	H940	400	300
456	Turner	H940	400	45

e.g. Room H940 has been improved, it is now of RSize = 500. For updating a single entity, we have to update all other columns where room=H940.

Enforcing Constraints

- With the simple file solution there is no way to enforce integrity constraints on the data. In other words people can put bad data into the text file.
- In contrast, a DBMS allows us to enforce all kinds of constraints. This really helps (but does not guarantee) that our data is correct.

A typo gives Roberta Wickham a GPA of 44.00



STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	44.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Scalability

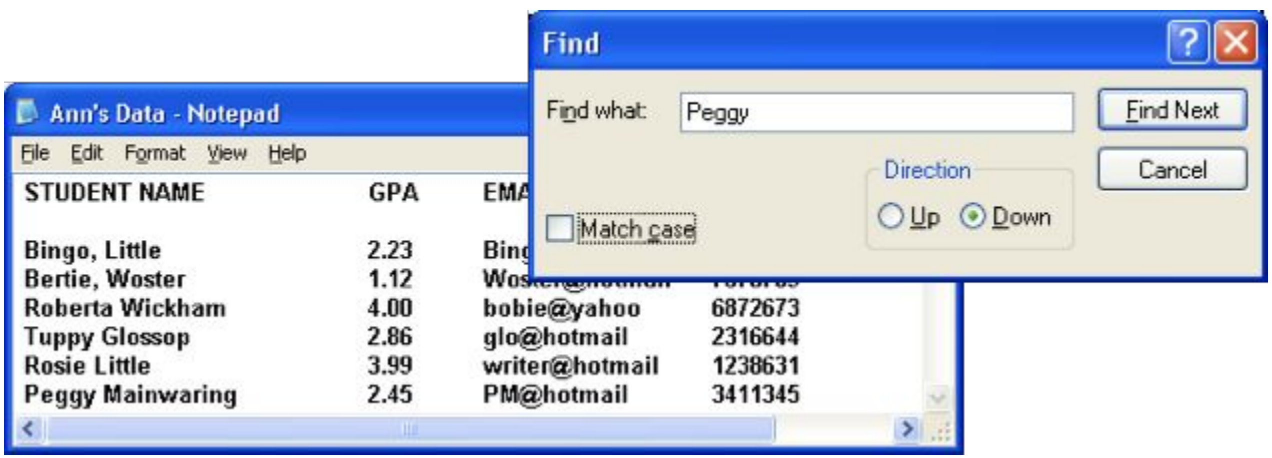
- The simple file solution might work for small datasets. What happens when we have big datasets?
- Most real world datasets are so large that we can only have a small fraction of them in main memory at any time, the rest has to stay on disk.
- Even if we had lots of main memory, with 32 bit addressing we can only refer to 4GB of data!

STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	4.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Query Expressiveness

- The simple file solution would allow me to search for keywords or certain numbers (slowly).

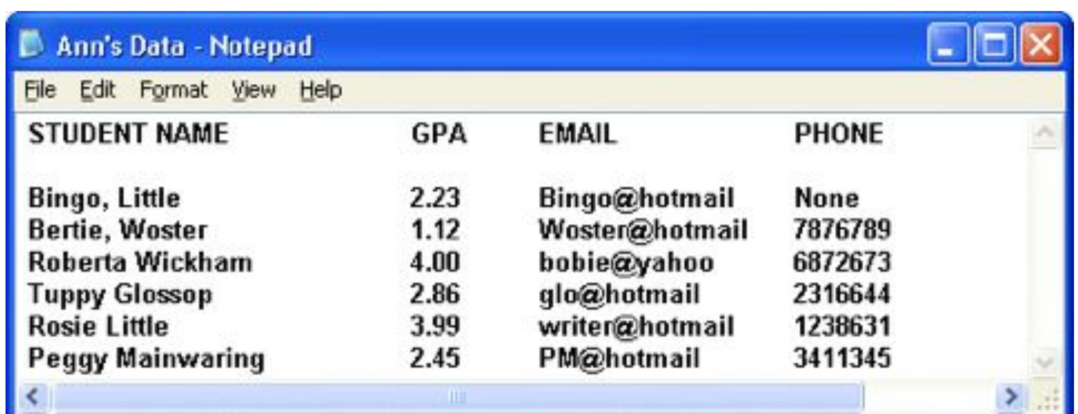
- With a DBMS I can search with much more expressive queries. For example I can ask.. "*Find all students whose GPA is greater than 2.5, and who don't own a phone*" or "*what is the average GPA of the students*"



Query Expressiveness II

- I could write some program that might allow more expressive queries on my text file, but it would tied into the structure of my data and the operating system etc..

- With a DBMS we are completely isolated from the physical structure of our data. If we change the structure of our data (by adding a field, for example) or moving from a PC to a Mac, nothing changes at the front end!



Ann's Data - Notepad

File Edit Format View Help

STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	4.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Different Views

- With a DBMS I can arrange for different people to have different views of the data. For example, I can see everything, a student can see only his/her data, the TA can see…

STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	4.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Concurrency

- Suppose I leave my text file on UNIX account, and I log in and begin to modify it at the same time my TA is modifying it!
- A DBMS will automatically make sure that this kind of thing cannot happen.

STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	4.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Security

- Suppose I leave my text file on UNIX account, and a student hacks in and changes their grades…

- A DBMS will allow multiple levels of security.

Ann's Data - Notepad

File Edit Format View Help

STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	4.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Crash Recovery

- Suppose I am editing my text file and the system crashes!

- A DBMS is able to guarantee 100% recovery from system crashes.

STUDENT NAME	GPA	EMAIL	PHONE
Bingo, Little	2.23	Bingo@hotmail	None
Bertie, Woster	1.12	Woster@hotmail	7876789
Roberta Wickham	4.00	bobie@yahoo	6872673
Tuppy Glossop	2.86	glo@hotmail	2316644
Rosie Little	3.99	writer@hotmail	1238631
Peggy Mainwaring	2.45	PM@hotmail	3411345

Database Management Systems

- What is a DBMS?
  - A “big” program written by someone else that allows us to manage efficiently a large database and allows it to persist over long periods of time.
    - Built to eliminate insert, delete, update anomalies

A **Database Management System (DBMS)** is a piece of software designed to store and manage data.

Relational Model Example

- Consider building a course management system (**CMS**):
  - Students
  - Courses
  - Professors

Entities

  - Who takes what
  - Who teaches what

Relationships

Data models

- A **data model** is a collection of concepts for describing data
    - The relational model of data is the most widely used model today
        - Main Concept: the *relation*- essentially, a table
- A **schema** is a description of a particular collection of data, **using the given data model**
    - E.g. every *relation* in a relational data model has a *schema* describing types, etc.

Modeling the Course Management System

- *Logical Schema*
  - Students(sid: *string*, name: *string*, gpa: *float*)
  - Courses(cid: *string*, cname: *string*, credits: *int*)
  - Enrolled(sid: *string,* cid*: string,* grade*: string*)

Relations

sid	**Name**	**Gpa**
101	Bob	3.2
123	Mary	3.8

Students

sid	**cid**	**Grade**
123	564	A

Enrolled

cid	**cname**	**credits**
564	564-2	4
308	417	2

Courses

Modeling the Course Management System

- *Logical Schema*
  - Students(sid: *string*, name: *string*, gpa: *float*)
  - Courses(cid: *string*, cname: *string*, credits: *int*)
  - Enrolled(sid: *string,* cid*: string,* grade*: string*)

sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

Students

Corresponding *keys*

sid	cid	Grade
123	564	A

Enrolled

cid	cname	credits
564	564-2	4
308	417	2

Courses

Other Schemata…

- *Physical Schema*: describes data layout
  - Relations as unordered files
  - Some data in sorted order (index)
- *Logical Schema:* Previous slide
- *External Schema*: (Views)
  - Course_info(cid: *string*, enrollment: *integer*)
  - Derived from other tables

Administrators

Applications

Data independence

<u>Concept:</u> Applications do not need to worry about *how the data is structured and stored*

<u>Logical data independence:</u> protection from changes in the *logical structure of the data*

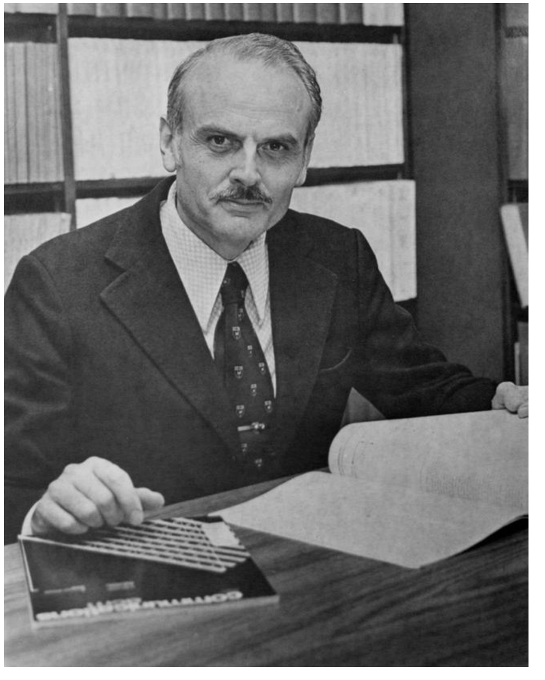
I.e. should not need to ask: can we add a new entity or attribute without rewriting the application?

<u>Physical data independence:</u> protection from *physical layout changes*

I.e. should not need to ask: which disks are the data stored on? Is the data indexed?

One of the most important reasons to use a DBMS

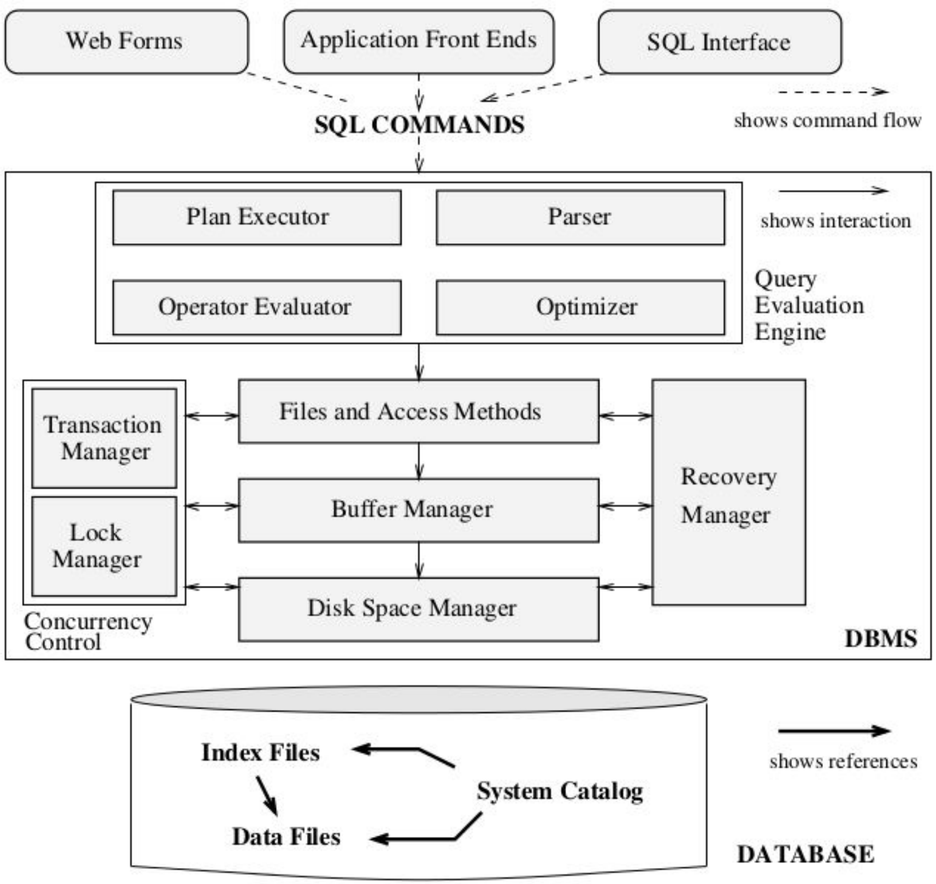
Relational Model in a nutshell



Invented by E. F. Codd in 1970s at IBM Research in CA

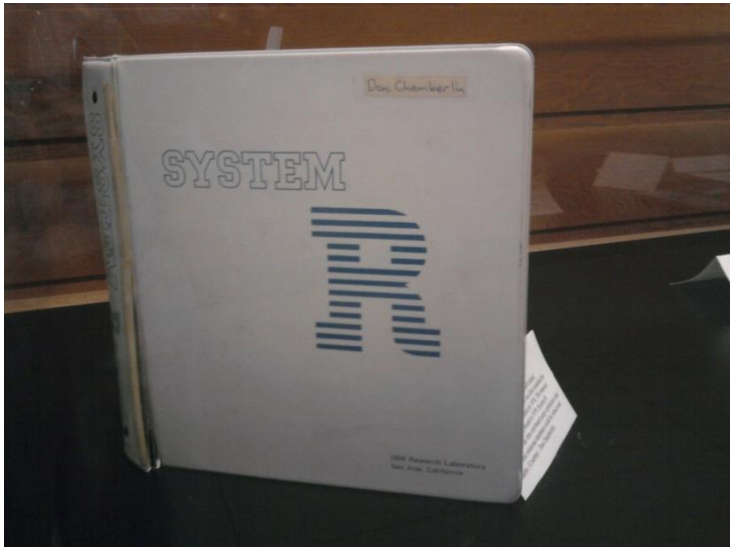
Relational Model in a nutshell

A software system to implement the relational model, i.e., enable users to manage data stored as relations



Relational Model in a nutshell

First RDBMSs: System R (IBM) and Ingres (Berkeley) in 1970s



A rare photo of the original System R manual

Mike Stonebraker won the Turing Award in 2015!

Relational Model in a nutshell

RDBMS software is now a USD 40+ billions/year industry!

Numerous open source RDBMSs also popular

People still start companies about what are basically RDBMSs!

Timeline of Databases

- **1960s** – hierarchical databases which provided support for concurrency, recover, and fast access.
- **1970-1972** - Edgar Codd who was working at IBM proposed the ‘relational database model’. Provided support for more reliability, less redundancy, more flexibility, etc.
- **1970s** – two major RDBMS prototypes were proposed: Ingres and System R
- **Mid 1970s** – A DB model called Entity –Relationship(ER) was proposed
- **1980s** – Structured Query Language (SQL) became standard querying language.
- **Late 1980s - 1990s** – Parallel and distributed databases

Challenges with Many Users

- Suppose that our CMS application serves 1000’s of users or more- what are some **challenges?**

  - Security: Different users, different roles

    We won’t look at too much in this course, but is extremely important

  - Performance: Need to provide concurrent access

    Disk/SSD access is slow, DBMS hide the latency by doing more CPU work concurrently

  - Consistency: Concurrency can lead to update problems

    DBMS allows user to write programs
    as if they were the **only** user

Transactions

- A key concept is the **transaction (TXN)**: an **atomic** sequence of db actions (reads/writes)

<u>Atomicity</u>: An action either completes *entirely* or *not at all*

Acct	Balance
a10	20,000
a20	15,000

Transfer $3k from a10 to a20:

1. Debit $3k from a10
2. Credit $3k to a20

- Crash before 1,
- After 1 but before 2,
- After 2.

Acct	Balance
a10	17,000
a20	18,000

DB Always preserves atomicity!

Written naively, in which states is **atomicity** preserved?

Transactions

- A key concept is the **transaction (TXN)**: an **atomic** sequence of db actions (reads/writes)
  - If a user cancels a TXN, it should be as if nothing happened!

- Transactions leave the DB in a **consistent** state
  - Users may write integrity constraints, e.g., 'each course is assigned to exactly one room'

However**,** note that the DBMS does not understand the *real* meaning of the constraints– consistency burden is still on the user!

Atomicity: An action either completes *entirely* or *not at all*

Consistency: An action results in a state which conforms to all integrity constraints

Challenge: Scheduling Concurrent Transactions

- The DBMS ensures that the execution of $\{T_1,\ldots,T_n\}$ is equivalent to some **serial** execution
- One way to accomplish this: **Locking**
  - Before reading or writing, transaction requires a lock from DBMS, holds until the end
- **Key Idea***:* If T_i wants to write to an item x and T_j wants to read x, then T_i, T_j **conflict***.* Solution via locking:
  - only one winner gets the lock
  - loser is blocked (waits) until winner finishes

A set of TXNs is **isolated** if their effect is as if all were executed serially

What if T_i and T_j need X and Y, and T_i asks for X before T_j, and T_j asks for Y before T_i?
-> *Deadlock!* One is aborted…

All concurrency issues handled by the DBMS…

Ensuring Atomicity & Durability

- DBMS ensures **atomicity** even if a TXN crashes!
- One way to accomplish this: **Write-ahead logging (WAL)**
- **Key Idea***:* Keep a log of all the writes done.
  - After a crash, the partially executed TXNs are undone using the log

Write-ahead Logging (WAL): Before any action is finalized, a corresponding log entry is forced to disk

We assume that the log is on "stable" storage

All atomicity issues also handled by the DBMS…

Summary of DBMS

- DBMS are used to maintain, query, and manage large datasets.
  - Provide concurrency, recovery from crashes, quick application development, integrity, and security
- Key abstractions give **data independence**
- DBMS R&D is one of the broadest, most exciting fields in CS. **Fact!**