Welcome to

CS 460: Introduction to Database Systems

https://midas.bu.edu/classes/CS460/

Instructor: Manos Athanassoulis
email: mathan@bu.edu
Today

big data

data-driven world

databases & database systems

when you see this, I want you to speak up!
[and you can always interrupt me]

no smartphones

no laptop
Big Data

marketing term ...

but ...

science / government / business / personal data

exponentially growing data collections

So, it is all good!
How big is “Big”?  

Every day, we create 2.5 exabytes* of data — 90% of the data in the world today has been created in the last two years alone.  

[Understanding Big Data, IBM]

*exabyte = $10^9$ GB
Using Big Data

- experimental physics (IceCube, CERN)
- biology
- neuroscience

- data mining business datasets
- machine learning for corporate and consumer

- data analysis for fighting crime

... are only some examples
Data-Driven World

Big Data V’s

Volume

Velocity

Variety

Veracity

Information is transforming traditional business.

[“Data, data everywhere”, Economist]
Data-Driven World

Discovery

Exploration

Data-to-Insight

Automated Decisions

Behind all these: use & manage data
we live in a *data-driven* world

CS460 is about the *basics* for *storing, using, and managing* data
your lecturer (that’s me!)

Manos Athanassoulis

name in greek: Μάνος Αθανασούλης

grew up in Greece
enjoys playing basketball and the sea

BSc and MSc @ University of Athens, Greece
PhD @ EPFL, Switzerland
Research Intern @ IBM Research Watson, NY
Postdoc @ Harvard University

some awards:
SNSF Postdoc Fellowship
IBM PhD Fellowship
Best of SIGMOD 2017, VLDB 2017

http://cs-people.bu.edu/mathan/
Office: MCS 106
Office Hours: M/W before class
your awesome TA

Dimitris Staratzis
grad student in DB

dstara@bu.edu
Data

to make data usable and manageable

we organize them in collections
Databases

a large, integrated, structured collection of data

intended to model some real-world enterprise

Examples: a university, a company, social media

University: students, professors, course
what is missing?
-- how to connect these?
-- enrollment, teaching

What about a company? What about social media?
Database Systems

a.k.a. database management systems (DBMS)
a.k.a. data systems

Sophisticated pieces of software...

... which store, manage, organize, and facilitate access to my databases ...

... so I can do things (and ask questions) that are otherwise hard or impossible
“relational databases are the foundation of western civilization”

Bruce Lindsay, IBM Research
ACM SIGMOD Edgar F. Codd Innovations award 2012
Ok but what really IS a database system?

Is the WWW a DBMS?

Is a File System a DBMS?

Is Facebook a DBMS?
Is the WWW a DBMS? Not really!

Fairly sophisticated search available
web crawler *indexes* pages for fast search

.. but

data is *unstructured* and *untyped*
not well-defined “correct answer”
cannot update the data
freshness? consistency? fault tolerance?

web sites *use* a *DBMS* to provide these functions
e.g., amazon.com (Oracle), facebook.com (MySQL and others)
“Search” vs. Query

What if you wanted to find out which actors donated to the first Barrack Obama’s presidential campaign 11 years ago?

Try “actors donated to obama” in your favorite search engine.
“Search” vs. Query

“Search” can return only what’s been “stored”

E.g., best match at Google:
A “Database Query” Approach

where can we find data for “all actors”?  

where can we find data for “all donations”?
A “Database Query” Approach
“IMDB Actors” JOIN “OpenSecrets”

<table>
<thead>
<tr>
<th>Contributor</th>
<th>Employer</th>
<th>Date</th>
<th>Amount</th>
</tr>
</thead>
<tbody>
<tr>
<td>ROCK, CHRIS MR</td>
<td>ACTOR</td>
<td>4/20/07</td>
<td>$9,200</td>
</tr>
<tr>
<td>NEW YORK,NY 10019</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOUGLAS, MICHAEL</td>
<td>ACTOR/ PRODUCER</td>
<td>3/30/07</td>
<td>$4,600</td>
</tr>
<tr>
<td>UNIVERSAL CITY, CA 91608</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DOUGLAS, MICHAEL</td>
<td>ACTOR/ PRODUCER</td>
<td>3/30/07</td>
<td>$2,300</td>
</tr>
<tr>
<td>UNIVERSAL CITY, CA 91608</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ROCK, CHRIS MR</td>
<td>ACTOR</td>
<td>4/20/07</td>
<td>$2,300</td>
</tr>
<tr>
<td>NEW YORK,NY 10019</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARIDES, GEORGIA</td>
<td>ACTOR</td>
<td>5/18/07</td>
<td>$1,000</td>
</tr>
<tr>
<td>NEW YORK,NY 10017</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CARTER COVINGTON, CLAUDIA</td>
<td>ACTORS THEATRE PART TIME/ACTOR/NEW</td>
<td>5/20/08</td>
<td>$1,000</td>
</tr>
<tr>
<td>CHARLOTTE,NC 28207</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>FOX, RICK</td>
<td>ACTOR/PRODUCER</td>
<td>6/16/08</td>
<td>$1,000</td>
</tr>
<tr>
<td>ENCINO, CA 91316</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HILDRETH, THOMAS W</td>
<td>ACTOR</td>
<td>9/29/08</td>
<td>$1,000</td>
</tr>
<tr>
<td>LOS ANGELES, CA 90068</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>RENNER, CARL</td>
<td>ACTOR/BESSONE@ROADRUNNER.COM</td>
<td>8/28/08</td>
<td>$1,000</td>
</tr>
<tr>
<td>BEVERLY HILLS, CA 90210</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>SIMMONS, HENRY</td>
<td>ACTOR</td>
<td>6/4/07</td>
<td>$1,000</td>
</tr>
<tr>
<td>WEST HOLLYWOOD, CA 90046</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Is a File System a DBMS?

Not really!

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 of your changes survive?

A) All  B) None  C) All Since last save  D) ???
Is Facebook a DBMS?

Is the data structured & typed?

Does it offer well-defined queries?

Does it offer properties like “durability” and “consistency”?

Not really!

Facebook is a data-driven company that uses several database systems (>10) for different use-cases (internal or external).
Why take this class?

*computation* to *information*

corporate, personal (web), science (big data)

database systems *everywhere*

data-driven world, data companies

DBMS: much of CS as a practical discipline

languages, theory, OS, logic, architecture, HW
CS460 in a nutshell

*model*
data representation model

*query*
query languages – ad hoc queries

*access* (concurrently multiple reads/writes)
ensure *transactional* semantics

*store* (reliably)
maintain *consistency/semantics in failures*
A “free taste” of the class

data modeling
query languages
concurrent, fault-tolerant data management
DBMS architecture

Coming in next class

Discussion on database systems designs
Componets of a "classic" DBMS

- Query Compiler
- Transaction Manager
- Schema Manager
- Execution Engine
- Logging/Recovery
- Concurrency Control

DBMS: a set of cooperating software modules
Describing Data: Data Models

**data model**: a collection of concepts describing data

**relational model** is the most widely used model today

key concepts

**relation**: basically a table with rows and columns

**schema**: describes the columns (or fields) of each table
Schema of “University” Database

**Students**

\[ \text{sid}: \text{string, name}: \text{string, login}: \text{string, age}: \text{integer, gpa}: \text{real} \]

**Courses**

\[ \text{cid}: \text{string, cname}: \text{string, credits}: \text{integer} \]

**Enrolled**

\[ \text{sid}: \text{string, cid}: \text{string, grade}: \text{string} \]
Levels of Abstraction

what the users see

what is the data model

how the data is physically stored
e.g., files, indexes

External Schema 1

External Schema 2

Conceptual Schema

Physical Schema
Schemas of “University” Database

Conceptual Schema

Students

- **sid**: string, **name**: string, **login**: string, **age**: integer, **gpa**: real

Courses

- **cid**: string, **cname**: string, **credits**: integer

Enrolled

- **sid**: string, **cid**: string, **grade**: string

Physical Schema

- relations stored in heap files
- indexes for sid/cid
Schemas of “University” Database

External Schema

a “view” of data that can be derived from the existing data

example: Course Info

Course_Info (cid: string, enrollment: integer)
Data Independence

Abstraction offers “application independence”

**Logical data independence**
Protection from changes in *logical* structure of data

**Physical data independence**
Protection from changes in *physical* structure of data

Q: Why is this particularly important for DBMS?

Applications can treat DBMS as black boxes!
Queries

”Bring me all students with gpa more than 3.0”

“SELECT * FROM Students WHERE gpa>3.0”

SQL – a powerful **declarative** query language

treats DBMS as a black box

What if we have multiples accesses?
Concurrency Control

*multiple users/apps*

**Challenges**

*how frequent access to slow medium*

how to keep CPU busy

how to avoid *short jobs* waiting behind *long ones*

e.g., *ATM withdrawal* while summing all *balances*

*interleaving actions of different programs*
Concurrency Control

Problems with *interleaving* actions of diff. programs

**Bad interleaving:**

- Savings \(-=\) 100
- Print balances
- Checking \(+=\) 100

*Printout is missing 100$ !"
Concurrency Control

Problems with *interleaving* actions of diff. programs

What is a correct interleaving?

- Savings $\leftarrow 100$
- Checking $\rightarrow 100$
- Print balances

How to achieve this interleaving?
Scheduling Transactions

Transactions: atomic sequences of \textbf{Reads} & \textbf{Writes}

\[
T_{Bill} = \{ R_{1\text{ Savings}}, R_{1\text{ Checking}}, W_{1\text{ Savings}}, W_{1\text{ Checking}} \}
\]
\[
T_{Alice} = \{ R_{2\text{ Savings}}, R_{2\text{ Checking}} \}
\]

How to avoid previous problems?
Scheduling Transactions

All interleaved executions equivalent to a **serial**

All actions of a transaction executed **as a whole**

How to achieve one of these?
before an object is accessed a lock is requested
Locking

before an object is accessed a lock is requested
before an object is accessed a lock is requested
Locking

locks are held until the end of the transaction

[this is only one way to do this, called “strict two-phase locking”]
Locking

T_1 = \{ R_1_{\text{Savings}}, R_1_{\text{Checking}}, W_1_{\text{Savings}}, W_1_{\text{Checking}} \}
T_2 = \{ R_2_{\text{Savings}}, R_2_{\text{Checking}} \}

Both should lock Savings and Checking

What happens:
if T_1 locks Savings & Checking ?
T_2 has to wait
if T_1 locks Savings & T_2 locks Checking ?
we have a deadlock
How to solve deadlocks?

we need a mechanism to **undo**

also when a transaction is **incomplete**

*e.g.*, due to a crash

**what can be an undo mechanism?**

*log every action **before** it is applied!*
Transactional Semantics

Transaction: one execution of a user program
multiple executions → multiple transactions

Every transaction:

- **Atomic**: executed entirely or not at all
- **Consistent**: leaves DB in a consistent state
- **Isolated**: as if it is executed alone
- **Durable**: once completed is never lost
Transactional Semantics

Transaction: one execution of a user program
multiple executions $\rightarrow$ multiple transactions

Every transaction:
- **Atomic** “executed entirely or not at all”
- **Consistent** “leaves DB in a consistent state”
- **Isolated** “as if it is executed alone”
- **Durable** “once completed is never lost”
Who else needs transactions?

- lots of data
- lots of users
- frequent updates
- background game analytics

Scaling games to epic proportions,
by W. White, A. Demers, C. Koch, J. Gehrke and R. Rajagopalan
ACM SIGMOD International Conference on Management of Data, 2007
Only “classic” DBMS?

No, there is much more!

NoSQL & Key-Value Stores: No transactions, focus on queries

Graph Stores

Querying raw data without loading/integrating costs

Database queries in large datacenters

New hardware and storage devices

... many exciting open problems!
Next time in ...

CS 460: Introduction to Database Systems

Database Systems Architectures
Class administrativia
Class project administrativia
If you require additional accommodations please contact the Disability & Access Services office at aslods@bu.edu or 617-353-3658 to make an appointment with a DAS representative to determine which are the appropriate accommodations for your case.

Please be aware that accommodations cannot be enacted retroactively, making timeliness a critical aspect for their provision.

You can optionally choose to disclose this information to the instructor.