Models & Intro to DB Architectures
prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
welcome brave cs165 students!

42+44
NO LAPTOP/PHONE POLICY

class is based on participation!

we will bring a copy of the slides for every one in each time class so you can follow and keep notes

+ there is enough evidence that laptops and phones slow you down (check syllabus for more info)
applications

database kernel

algorithms/operators

data

data

data

cpu

memory

disk
a “simple” example

assume an array of $N$ integers:
find all positions where $value > x$

exist in all systems: sql, nosql, newsql

even the simplest tasks are actually far from trivial
no obvious solutions; just a taste of what to come
what we will do
design data structures and algorithms = access methods
study design tradeoffs with respect to modern hardware
application requirements & complete system design

next couple of weeks
very basics of data models and languages (today)
basics of db architectures (today next couple of classes)
column-stores and hardware-conscious designs
September 2, 2016 — October 2, 2016

**Milestone 1: Basic column-store**

The goal is to design and implement the basic functionality of a column-store with the ability to run single-table queries.
for project:
install MonetDB & PostgreSQL/MySQL and play with SQL

explain + SQL query to see query plans in MonetDB (use read-only mode)

repeat throughout the semester compare with your system in terms of performance
this is part of final deliverable

& several logistics and tools
(code.seas, gdb, perf, valgrind, testing infrastructure)

do not underestimate this…
what should I be doing?

do P0
register to Piazza and stay logged in
check syllabus/website carefully
check project timeline and plan around it
keep up with reading (goes fast)
register for notes today (tmr we will do assignments)

come to Labs and OH frequently
how to read research papers

1) abstract-intro-related work-conclusions
what is the problem
why is it important
why past solutions do not work
what is the core idea
what is success

2) core part-analysis
basic idea
what matters
any gaps?

3) follow a few citations and repeat

goal: by the end of the semester understand these papers fully
we want you to have fun!
data systems is an exciting field!

tell us how you are keeping up
tell us what you need to better follow the class
tell us your suggestions about how to improve the class
the evolution of data-driven applications
Anand Rajaraman

(1) use own data (recommendations)
(2) use public data (search)
(3) use social data (social networks)
(4) all of the above (all of the above)
(5) training data (machine learning)
food for (big data-driven) thought

the cyborg knows & manages all your models
food for (big data-driven) thought
the biggest transportations companies have no cars
the biggest data companies have no data
the biggest hotel companies have no hotels

more about this and other thoughts in our first brainstorming session (TBA)
logical design

physical design

system design
essential steps in using a database system

clean → schema → load → tune

query

experts/system admins

user/apps
relational model + SQL

create table for professors:

*create table* professors (id:integer, name: char(40), telephone: char(10), …)

*insert into* professors (76897689, “john smith”, …)

give me the names of all students:

*select* name *from* students *where* GPA>3.0
employee
(id:int, name:varchar(50), office:char(5), telephone:char(10), city:varchar(30), salary:int)

(1, name1, office1, tel1, city1, salary1)
(2, name2, office2, tel2, city2, salary2)
(3, name3, office3, tel3, city3, salary3)
(4, name4, office4, tel4, city4, salary4)
(5, name5, office5, tel5, city5, salary5)
(6, name6, office6, tel6, city6, salary6)
(7, name7, office7, tel7, city7, salary7)
(8, name8, office8, tel8, city8, salary8)
(9, name9, office9, NULL, city9, salary9)

SQL:insert into employee
(1, name1, office1, tel1, city1, salary1)

value does not exist

cardinality=9
give me all students enrolled in cs165

```
select student.name from student, enrolled, course where
    course.name="cs165" and enrolled.courseld=course.id and
    student.id=enrolled.studentId
```
enrolled
(studentId, courseId, ...)

student
(id, name, ...)

how do we join?
say schema about university db contains one table…

**AllData** (student ID, student name, student address, course name, grade, professor name, professor ID, professor telephone, …)

duplicates - tons of data - updates - but no joins
star schema

fact table
(id1, id2, ...)

dimension table 1
(id1, ...)

dimension table 2
(id2, ...)

...
snowflake schema
Alex Liu, class 2016
165/265 project
adaptive denormalization

1st prize in ACM SIGMOD undergrad research competition

Special Interest Group
on Management of Data
NORMALIZED DATA

good for updates, storage but we need joins

DENORMALIZED DATA

only fast scans but expensive to create, storage & updates
adaptive denormalization

continuously physically reorganize data based on incoming query patterns (joins)

denormalized fragments
queries only need to fast scan

possible denormalized space

normalized data
create table employee
(id:integer,
name:varchar(50) not null,
office:char(5),
phone:char(10),
city:varchar(30),
salary:integer,
primary key (id)
check (salary<100000))

must have a value
at most 5 chars
must be unique
must not become rich

when and how do we enforce constraints
aggregations

```
select max(GPA), avg(GPA), min(GPA)
from students
```

math

```
select R.a - R.b + R.c
from R
```

nested

```
select *
from R
where R.a IN (select b
  from S
  where C<10)
```

set ops

```
select * from R where a = 10
UNION
select * from B where b = 20
```
```sql
select avg(GPA), class, major
from students
where GPA > 3.0 and class > 1990
group by class, major
order by class
```
**Employee**
(id:int, name:varchar(50), office:char(5), telephone:char(10), city:varchar(30), salary:int)

**base table**

**Employee-Berlin-Manager**
select * from employee where city="berlin"

**view** to be used by managers in Berlin

**Employee-Berlin-All**
select id,name,city,office from employee where city="berlin"

**view** to be used by all employees in Berlin

how should we store views
why are models great?
other models?
it is summer 2017 - now you know all about data systems

you are building an **augmented reality startup** using Google Glass

people wearing Google Glass can tag places/objects - voice/image recognition works fine

tagging means assigning values, comments, etc to an object

you can then query this data - again assume voice recognition works fine and a black box translates natural language to SQL

**how does the schema of your app look like?** (tables, attributes, keys, relationships)
(assume a limited working environment/features, say walking around Harvard square/yard)

**describe 2 interesting queries in SQL**
q1: get all places where jenny said “awesome”

q2: get all users that like what I like and are close by

```
select user.name, user.location
from user, likes_object as L1, likes_object as L2
where L1.user_id=my_id and L1.object_id=L2.object_id and L2.user_id != my_id
and user.id=L2.user_id and close(user.location, my_location) = true
```

```
select object.location
from object, user
where user.name = “jenny” and user_id = L1.user_id
and comment.user_id = user.id and comment.text LIKE “%awesome%”
```
how do we store the object table?

what if we want to add another kind of object?

**object**
(id, name, location, telephone, date, url, color, taste, ...many more)

open research and business problem
design

- logical design
- physical design
- system design
declarative interface
ask what you want

so do db systems “just work”?
declarative interface
ask what you want

but … db cracking, adaptive* ideas

indexes/views/tuning knobs

db system
essential steps in using a database system

1. Clean
2. Schema
3. Load
4. Tune
5. Query

Experts/system admins

User/apps
Design

- Logical design
- Physical design
- System design

Next up: DB architectures 101
design/implement numerous possible algorithms + data representations

choose the best data source, algorithms and path for each query
\textbf{select} \ \text{min}(A) \ \textbf{from} \ R \ \textbf{where} \ B<10 \ \text{and} \ C<80
is it “good” to have modules
Notes to remember

models help create the right abstractions
models help create >> 1 applications over the same data
we first need to clean, structure and load data
data systems consist of software components
reading

textbook: chapters 1, 3 (-3.5), 5 (-5.8,-5.9)
intro + relational model + SQL

browse “the Fourth Paradigm”
readings for next 3 classes

**Architecture of a Database System** (Sections 1,2,3,4)
by J. Hellerstein, M. Stonebraker and J. Hamilton

**The Design and Implementation of Modern Column-store Database Systems**
by D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden
Models & Intro to DB Architectures

DATA SYSTEMS

prof. Stratos Idreos