class 3

Models & Intro to DB Architectures

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
welcome brave cs165 students!
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)
slides are not notes!

slides are mainly there to trigger discussion

note keeping is your task:
starting class 3 we will do collaborative note taking
Will cs165 help me with using (big, nosql, relational) data systems

Yes it will
but if that is your only goal,
there are easier ways to do this
database kernel

applications

sql

algorithm/operators

data

data

data

cpu

memory

disk
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
for the semester 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…
Read completely and more than once

Read intro/related work, browse main part

Read only if interested
From the syllabus: “In this class many of the reading assignments are recent research papers. Unless mentioned otherwise in class, you are not expected to read and understand all these research papers in extreme detail. The main purpose is for you to get exposed to recent ideas and concepts and get inspiration about new opportunities and what is coming in the future. We expect you to read and understand well the abstract, introduction and related work parts of all papers. For the rest of the material in a paper, i.e., the main technical part and the analysis we expect you to have a high level idea of what this does unless we explicitly cover in detail the exact techniques in class. Our goal is that by the end of the semester you will have enough background to be able to pick up any of these papers again and understand it fully! Of course if you want to discuss any of those papers in more detail we will happily do so during office hours.”
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
next few weeks:
(1) data models and languages (today)
(2) db architectures (today & next few classes)
(3) column-stores & hardware-conscious designs
Extra: 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

thousands of users - tons of data - frequent updates - tons of queries
analytics in the background, for game intelligence
= the ultimate data-driven challenge
a “simple” example

assume an array of \( N \) integers:
find all positions where \( \text{value} > x \)

qualifying positions

select operator

even the simplest tasks are actually far from trivial
no obvious solutions; just a taste of what to come
~1960s

philosophy/sciences

papyrus/paper

~2017
dbs vs OSs
dbs vs nosql

declarative/physical/logical independence/recovery/consistency

history/timeline

dbs vs hand code
As apps become more complex, as apps need to be more scalable, the need for newSQL to db abstractions becomes clear. Declarative processing and optimization of memory hierarchy and I/O vs CPU modeling involve tradeoffs in data layout concepts.

...
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

Anand Rajaraman
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

1. clean
2. schema
3. load
4. tune
5. query

For experts/system admins:

For 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`
The schema and data for the `employee` table are as follows:

**Schema**

```
employee
(id: int, name: varchar(50), office: char(5),
 telephone: char(10), city: varchar(30), salary: int)
```

**Data**

```
(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)

The value for `office9` is `NULL`, so the `insert` operation fails.

Cardinality: 9

A message appears stating: "value does not exist".
give me all students enrolled in cs165

```
select student.name from student, enrolled, course where
course.name = 'cs165' and enrolled.courseId = 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

query: no joins
try to insert data
now try to update professor data
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
create table employee
(id:integer,
 name:varchar(50) not null, ← must have a value
 office:char(5), ← at most 5 chars
 telephone:char(10),
 city:varchar(30),
 salary:integer,
 primary key (id) ← must be unique
 check (salary<100000)) ← must not become rich

when and how do we enforce constraints
more SQL examples

aggregations

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

math

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

nested

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

set ops

```sql
select * from R where a = 10
UNION
select * from B where b = 20
```
select avg(GPA), class, major
from students
where GPA>3.0 and class>1990
group by class, major
order by class
base table

**Employee**
(id:int, name:varchar(50), office:char(5), telephone:char(10), city:varchar(30), salary:int)

view to be used by managers in Berlin

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

view to be used by all employees in Berlin

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

Why is this useful
How should we store views
why are data models great?

allow for abstraction
other models?

graph, RDF,
it is summer 2018 - 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)

**describe 2 interesting queries in SQL**

user(id,first,last,dob,curlocation,…)

comment(id,userld,objectld,text)

object(id,location, all attributes of objects with nulls: telephone, year built,
or only common attributes in object and store rest in extra tables with objld as both primary and foreign key)

likesC(userId,commentId)

likesO(userId,objID)
a possible example

q1: get all places where jenny said "awesome"

```
select object.location
from object, user
where user.name = "jenny" and comment.user_id=user.id and comment.text LIKE "%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,mylocation)=true
```

tell them about **dictionary compression** - you do not do it in the schema - system does it underneath - better not let the designer worry about low level details - not always possible

**someone do an SQL query:**

select o.location from O,U where user.name = jenny and comment.user_id=user.id and comment.text LIKE "%awesome%"

what the plan should look like? first the selection on comment or the join?

select names,location from U, likes_O as L1, likes_O 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,mylocation)=true
we are going to have TONS of objects! Talk about nodal
declarative interface
ask what you want

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

indexes/views/tuning knobs

but ... db cracking, adaptive* ideas

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 best possible storage/access level with a specific API
decision level
whenever we make decisions - things can go wrong
parse like compiler flex, yacc, language grammar etc
check catalog: maintain all meta, tables columns, — access control : am I allowed to see this data etc
who decides how to store the data? based on what?
simple listen send out with buffer

admission control - memory load etc simple number restriction - assign priorities etc…
diff between program and thread -
thread pool to avoid creating and destroying threads - databases had their own thread support - now use OS
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

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

Browse: “the Fourth Paradigm”
readings for next few classes

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

Read: **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