class 3

SQL & intro to db architectures

prof. Stratos Idreos

HTTP://DASLAB.SEAS.HARVARD.EDU/CLASSES/CS165/
welcome brave cs165 students!
35+62
Laura Haas

Data Systems Researcher
Director of IBM Research’s Accelerated Discovery Lab & Harvard alumna

October 5

a 1 hour discussion with students will follow after class

1 hour wics meeting 3:30-4:30

The Power Behind the Throne: Information Integration in the Age of Data-Driven Discovery
Peter Haas

Data Systems Researcher

IBM Almaden Research Center
& Harvard alumnus, class 78

October 5, 3:30pm

Balancing Recency and Continuity in Massive Scale Dynamic Interaction Graphs
Nga Tran

Data Systems Researcher

Head of Optimizer Group at HP Vertica

Nov 9
when should I start my project?

next week

in the meantime:
1) play with code base & tools
2) linked list, binary tree
3) MonetDB, PostgreSQL
which tests should I pass?

we care about DSL tests
by Wed it should be obvious why

**leaderboard** will test all DSL tests
so you will know exactly your status at any time

more tests soon
how much should I optimize?

enough...e.g., see m3

we expect a generally elegant design & implementation

often OH & sections

rule of thumb: at least 2 areas
starting today we will do **collaborative note taking:**
http://tinyurl.com/cs165-notes

ping for Wasay for access

slides are not notes!

**How:**
1) register for a class or two
2) take notes during class and put them in later on
3) edit notes

“your name: note”
assume an array of $N$ integers:
find all positions where $value > x$

not as simple as it looks…
logical design

physical design

system design
today
essential steps in using a database system

- clean
- schema
- load
- tune

Experts/system admins

User/apps

query
relational model + SQL

database

**professors**
(id, name, ...)

table/relation

key

**courses**
(id, name, profId, ...)

column/attribute

**students**
(id, name, ...)

table/relation
create table for professors:

`create table professors (id:integer, name: char(40), telephone: char(10), ...)`
create table for professors:

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

```sql
insert into professors (76897689, “john smith”, …)
```
relational model + SQL

database

- **professors** (id, name, …)
  - key
  - table/relation

- **courses** (id, name, profId, …)
  - column/attribute

- **students** (id, name, …)
  - table/relation

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
create table for professors:

\textbf{create table} professors (id:integer, name: char(40), telephone: char(10), …)

\textbf{insert into} professors (76897689, “john smith”, …)

give me the names of all students:

\textbf{select} name \textbf{from} students \textbf{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)

value does not exist

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

cardinality=9
relational model + SQL

give me all students enrolled in cs165
relational model + SQL

database

**professors**
(id, name, …)

**courses**
(id, name, profId, …)

**students**
(id, name, …)

**enrolled**
(studentId, courseId, …)

give me all students enrolled in cs165
relational model+SQL

give me all students enrolled in cs165
relational model + SQL

give me all students enrolled in cs165

\[
\text{select student.name from students, enrolled, courses where courses.name=“cs165” and enrolled.courseld=course.id and student.id=enrolled.studentId}
\]
relational model + SQL

give me all students enrolled in cs165

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

students
(id, name, ...)

how do we join?
Normalization

Say schema about university db contains one table…

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

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,\ldots)

dimension table 1
(id1,\ldots)

dimension table 2
(id2,\ldots)

...
snowflake schema

...
create table employee
(id:integer,
 name:varchar(50) not null,
 office:char(5),  \textbf{at most 5 chars}
 telephone:char(10),
 city:varchar(30),
 salary:integer,
 primary key (id)
 check (salary<100000)) \textbf{must be unique}

\textbf{must have a value}

\textbf{must not become rich}
create table employee
(id:integer,  
name:varchar(50) not null,  
office:char(5),  
check (salary<100000))

can be null

must have a value

at most 5 chars

must be unique

must not become rich

when and how do we enforce constraints
more SQL examples

aggregations

\[
\text{select max(GPA), avg(GPA), min(GPA)}
\]
from students

math

\[
\text{select } R.a - R.b + R.c
\]
from R

nested

\[
\text{select *}
\]
from R
where R.a IN (select b
from S
where C < 10)

set ops

\[
\text{select * from R where } a = 10
\]
UNION
\[
\text{select * from B where } b = 20
\]
select avg(GPA), class, major
from students
where GPA>3.0 and class>1990
group by class, major
order by class
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"
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"
**design**

- **logical design**
  - physical/logical independence

- **physical design**
  - app/user: no need to know how data is stored/accessed

- **system design**
  - we can safely change lower layers
it is summer 2015 - 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
a possible example

q1: get all places where jenny said “awesome”

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

```
comment
(id, user_id, object_id, text, ...)

likes_comment
(user_id, comment_id)

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

likes_object
(user_id, object_id,)

user
(id, name, location, device, ...)

trust
(user_id, user_id)
```
a possible example

q1: get all places where jenny said “awesome”

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

```
select object.location
from object, user
where user.name = "jenny" and
    comment.user_id=user.id and
    comment.text LIKE "%awesome%"
```
q1: get all places where Jenny said “awesome”

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

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,mylocation)=true
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)
logical design

physical design

system design
declarative interface
ask what you want

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

indexes/views/tuning knobs

db system
declarative interface
ask what you want

indexes/views/tuning knobs

but … db cracking, adaptive* ideas

db system
essential steps in using a database system

clean → schema → load → tune

query

experts/system admins

user/apps
design

next up: db architectures 101
declarative interface
ask what you want

the db system decides how to “best store and access data”
design/implement numerous possible algorithms + data representations

choose the best data source, algorithms and path for each query
**select** \( \text{min}(A) \) **from** \( R \) **where** \( B < 10 \) and \( C < 80 \)
applications

parser
optimizer
execution
storage

in/out
admission

sql

database kernel
Applications

Database kernel

Parser

In/out

Optimizer

Thread pool

Execution

Transactions

Storage

Buffer pool

CPU

Memory

Disk
is it “good” to have modules
diagram of query plan and database kernel
**SQL Query**

```sql
select name
from student
where GPA > 3.0
```

**Logical Plan**

1. `select GPA > 3.0` from `student(id, name, GPA, address, class,...)`
2. `project name`
3. `result`

**Question:** Should we scan all the data?
\textbf{select} name \\
\textbf{from} student \\
\textbf{where} GPA > 3.0

\textbf{physical plans}
\[ \text{select} \ \text{avg}(\text{GPA}) \\
\text{from} \ \text{student} \\
\text{where} \ \text{class}=2017 \]
give me all students enrolled in cs165

select student.name
from students, enrolled, courses
where courses.name="cs165"
and enrolled.courseld=course.id
and student.id=enrolled.studentId
good plan
select student.name
from students, enrolled, courses
where courses.name="cs165"
and enrolled.courseId=course.id
and student.id=enrolled.studentId
\textbf{select} \textbf{min}(A) \textbf{from} R \textbf{where} B<10 \textbf{and} C<80
concurrency

how many queries should a db run in parallel and how

reads - writes
concurrency

transactions
ACID properties
locks

Jim Gray, IBM, Tandem, DEC, Microsoft
ACM Turing award
ACM SIGMOD  Edgar F. Codd Inovations award 1993
recovery
what should happen if something fails during a query?
reads - writes
**classic example**

joe owes mike 100$

both joe and mike have a Bank of Bla account

**possible actions**

1) read mike; 2) mike+100; 3) write new mike;
recovery
logs
write ahead
replay
checkpoints

C Mohan, IBM
ACM SIGMOD  Edgar F. Codd Inovations award 1993
Can DBAs make wrong decisions? Can optimizers make wrong decisions?

db kernel

- tuning
- optimizer
- execution
- storage
Architecture of a Database System  (Sections 1,2,3,4)  
by J. Hellerstein, M. Stonebraker and J. Hamilton
readings for next few classes

The Design and Implementation of Modern Column-store Database Systems (Sections: all -4.6 & 4.8) by D. Abadi, P. Boncz, S. Harizopoulos, S. Idreos, S. Madden

IEEE Data Engineering Bulletin, 35(1), March 2012
Special Issue on Column-stores (9 short overview papers)

next class we start discussing data layouts and column-stores
SQL & intro to db architectures

DATA SYSTEMS

prof. Stratos Idreos

HARVARD
School of Engineering and Applied Sciences