Introduction

Introduction to Databases
CompSci 316 Spring 2020

Welcome to
CompSci 316: Introduction to Database Systems!!
Spring 2020

About us: instructor

• Instructor: Sudeepa Roy
  • At Duke CS since Fall 2015
  • Member of “Duke Database Devils” a.k.a. the database research group
  • PhD. UPenn, Postdoc: U. of Washington
  • Research interests:
    • “data”
    • data management, database theory, data analysis, data science, causality and explanations, uncertain data, data provenance, crowdsourcing, ....

Meet your grad TAs

• Tiangang Chen
  • Duke CS MS student
  • Runs a half-marathon every year!

• Yuchao Tao
  • Duke CS PhD student
  • Loves cats!

Meet your UTAs

• Runxin (Rebecca) Wang
  • Duke CS major
  • Loves hiking and bubble tea!

• Jane Li
  • Duke CS major, minor in Visual Media studies
  • Loves hiking and bubble tea!

• David Chen
  • Duke CS major, minor in Visual Media studies
  • Interested in UX/UI, front-end development, and project management
  • Has two dogs and a hamster!

What are the goals of this course?

• Learn about “databases” or data management
Why do we care about data? (easy)

How big data can help find new mineral deposits

… The three years of gathering and analyzing data culminated in what U.S. Sailing calls their “Rio Weather Playbook,” a study of critical information about each of the seven courses only available to the U.S. team —

FiveThirtyEight, “Will Data Help U.S. Sailing Get Back on the Olympic Podium?”
Aug 15, 2016

Wait… don’t we need to take a Machine Learning or Stat course for those things? Yes, but...

Data =
Money
Information
Power
Fun
in
Science, Business,
Politics, Security
Sports, Education, …

Also think about building a new App or website based on data from scratch

• E.g., your own version of book purchase platform (like a mini-Amazon)
• Large data! (think about all books in the world or even in English)

• How do we start?

* You are going to do something similar in the course project!

Who are the key people?
What should the user be able to do?

What should the platform do?

What are the desired and necessary properties of the platform?
That was the design phase (a basic one though)

How about C++, Java, or Python? On data stored in large files

Sounds simple!

• Text files – for books, customer, ...
• Books listed with title, author, price, and no. of copies
• Fields separated by #'s

Query by programming

Query by programming

Revisit: What are the desired and necessary properties of the platform?

Solution?

A DBMS takes care of all of the following (and more):

- Optimization
- Index
- Recovery
- Consistency
- Declarative

* We will learn these in the course!
DBMS helps the big ones!

Note: Not always the “standard” DBMS (called Relational DBMS), but we need to know pros and cons of all alternatives

CompSci 316 gives an intro to DBMS

• How can a user use a DBMS (programmer’s/designer’s perspective)
  • Run queries, update data (SQL, Relational Algebra)
  • Design a good database (ER diagram, normalization)
  • Use different types of data (Relational, XML, JSON)

• How does a DBMS work (system’s or admin’s perspective)
  • Storage, index
  • Query processing, join algorithms, query optimizations
  • Transactions: recovery and concurrency control

• Glimpse of advance topics and other DBMS
  • NOSQL, Spark (big data)
  • Data mining

Hands-on experience in class projects by building an end-to-end website or an app that runs on a database

Misc. course info

• Website: [https://www2.cs.duke.edu/courses/spring20/compsci316/](https://www2.cs.duke.edu/courses/spring20/compsci316/)
  • Course info; tentative schedule and reference sections in the book; lecture slides, assignments, help docs, …
• Programming: VM required, need significant programming on different platforms and languages
• Prerequisite: CompSci 320 (will need basic understanding of discrete maths, data structure, and algorithms) - or talk to us
• Q&A on Piazza
• Grades, sample solutions on Sakai
• Submissions on Gradescope and Gradiance
• Watch your email for announcements

Important: Grading

Absolute but adjustable grading
Guarantees:

- [90%, 100%] A / A / A+
- [80%, 90%) B / B / B+
- [70%, 80%) C / C / C+
- [60%, 70%) D

Class topper gets A+
• Scale will not go upwards but can get downwards (e.g., based on the class performance in the exams)

• We will give you a feedback on your approximate standing after the midterm.

Duke Community Standard

• See course website for link
• Group discussion for assignments is okay (and encouraged), but
  • Acknowledge any help you receive from others
  • Make sure you “own” your solution
  • All suspected cases of violation will be aggressively pursued

Course load

• (See course webpage for full details)

• Weekly (short) homework assignments (25%)
  • Each homework has same weight
  • Released on Tuesdays and due next Tuesday night (mostly)
  • Gradiance: immediately and automatically graded
  • Gradescope: written solution, manual grading
  • Gradescope: programming problems, immediate feedback, later also manual grading

• Midterm and final (20% each)
  • Open book, open notes
  • No communication/Internet whatsoever
  • Final is comprehensive, but emphasizes the second half of the course
Course load (contd.)

- **Course project (20%)**
  - Details to be given in the next 1-2 weeks

- **In-class quiz (5%)**
  - To review concepts right away in class – will be open for 5-10 mins
  - Will be announced at least one class in advance and on piazza
  - Each quiz: 50% for attempt on time and 50% for correct solution
  - Lowest score will be dropped (each quiz has same weight)

- **In-class labs (5%)**
  - Practice problems in class (both programming and conceptual) – each lab has the same weight
  - Will be announced at least one class in advance and on piazza
  - Due by the next day after class, 10% bonus points for finishing all problems in class correctly
  - TAs will be around to help you

Projects from past years

- RA: next-generation relational algebra interpreter
  - You may get to try it out for Homework #1!
- Managing tent shifts and schedules!
- Tutor-tutee matching
  - What’s in my fridge and what can I cook?
- HearSay: manage your own musics
- Dining at Duke (and deliver meals to students)
- National Parklopedia: a website to find information about national parks

- More examples later - but we expect you to be creative with a new idea!

Edgar F. Codd (1923-2003)

- Pilot in the Royal Air Force in WW2
- Inventor of the relational model and algebra while at IBM
- Turing Award, 1981

RDBMS = Relational DBMS

Tentative office hours schedules

- Locations: TBD.
- See the updated info on the webpage
- More office hours around Tuesday (hw due), but good to start early!

Let’s get started!

Relational Data Model

What is a good model to store data?

The famous “Beers” database

Bars each have an address
Bars serve beers at price “Y”
Drinkers like beers
Drinkers frequent bars “X” times a week
Drinkers each have an address
Beers each have a brewer

“Beers” as a Relational Database

<table>
<thead>
<tr>
<th>Bar</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Edge</td>
<td>108 Morris Street</td>
</tr>
<tr>
<td>Satisfaction</td>
<td>109 W. Main Street</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Beer</th>
<th>Brewer</th>
</tr>
</thead>
<tbody>
<tr>
<td>Budweiser</td>
<td>Anheuser-Busch Inc.</td>
</tr>
<tr>
<td>Corona</td>
<td>Grupo Modelo</td>
</tr>
<tr>
<td>Dixie</td>
<td>Dixie Brewing</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Drinker</th>
<th>Beer</th>
<th>Times a week</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amy</td>
<td>Corona</td>
<td></td>
</tr>
<tr>
<td>Ben</td>
<td>Budweiser</td>
<td></td>
</tr>
<tr>
<td>Dan</td>
<td>Corona</td>
<td></td>
</tr>
</tbody>
</table>

Relational data model

- A database is a collection of relations (or tables)
- Each relation has a set of attributes (or columns)
- Each attribute has a name and a domain (or type)
  - Set-valued attributes are not allowed
- Each relation contains a “set” of tuples (or rows)
  - Each tuple has a value for each attribute of the relation
  - Duplicate tuples are not allowed (Two tuples are duplicates if they agree on all attributes)
  - Ordering of rows doesn’t matter (even though output is always in some order)
- However, SQL supports “bag” or duplicate tuples (why?)
- Simplicity is a virtue
  - not a weakness!

Schema vs. instance

- Schema
  - Beer (name string, brewer string)
  - Serves (bar string, beer string, price float)
  - Frequent (drinker string, bar string, times_a_week int)
- Instance
  - Actual tuples or records

SQL: Querying a RDBMS

- SQL: Structured Query Language
  - Pronounced “S-QL” or “sequel”
  - The standard query language supported by most DBMS
  - First developed at IBM System R
  - Follows ANSI standards

Basic queries: SFW statement

- SELECT $A_1, A_2, ..., A_n$
FROM $R_1, R_2, ..., R_m$
WHERE condition

- SELECT, FROM, WHERE are often referred to as
SELECT, FROM, WHERE “clauses”

Example: reading a table

- SELECT *
FROM Serves

- Single-table query
  - WHERE clause is optional
  - * is a short hand for “all columns”
Example: selecting few rows

- SELECT beer AS mybeer
  FROM Serves
  WHERE price < 2.75

- SELECT beer
  FROM Serves
  WHERE bar = 'The Edge'

What does these return?

- SELECT list can contain expressions
- Can also use built-in functions such as SUBSTR, ABS, etc.
- String literals (case sensitive) are enclosed in single quotes
- "AS" is optional
- Do not want duplicates? Write SELECT DISTINCT beer ...

Example: Join

- Find addresses of all bars that ‘Dan’ frequents

• SELECT B.address
  FROM Bar B, Frequents F
  WHERE B.name = F.bar
  AND F.drinker = 'Dan'

- Okay to omit table_name in table_name.column_name if column_name is unique
- Can use “Aliases” for convenience
  • "Bar as B" or "Bar B"

Example: Join

- Find addresses of all bars that ‘Dan’ frequents

Which tables do we need?

How do we combine them?

Let’s try SQL in class!

(See how to access the pgweb interface for a small “Beers” database on the slides posted on the course website)

Next: semantics of SFW statements in SQL

Announcements (Tue, 01/09)

- You should be on Sakai, Piazza, Gradescope
  • If you are not there or recently enrolled, please contact the instructor
- You will receive instructions on installing the VM
  • Please follow Piazza posts, all notifications will be posted there and you should receive emails right away
- First homework to be released on next class Tuesday 01/14, due in a week
  • No in-class quiz or labs unless explicitly announced in the class before (and posted on Piazza)
Semantics of SFW

- SELECT $E_1, E_2, ..., E_n$
  FROM $R_1, R_2, ..., R_m$
  WHERE condition

  - For each $t_1$ in $R_1$:
    - For each $t_2$ in $R_2$, ...
      - Form cross-product of $R_1, ..., R_m$.
      - If condition is true over $t_1, t_2, ..., t_m$:
        - Apply “WHERE”
          - Only consider satisfying rows
          - Compute and output $E_1, E_2, ..., E_n$ as a row
          - Output the desired columns

Step 1: Illustration of Semantics of SFW

- NOTE: This is “NOT HOW” the DBMS outputs the result, but “WHAT” is outputs!
  Form Cross product of two relations

- SELECT $B$.address FROM Bar $B$, Frequents $F$
  WHERE $B$.name = $F$.bar AND $F$.drinker LIKE ‘Dan’

<table>
<thead>
<tr>
<th>Bar</th>
<th>address</th>
<th>drinker</th>
<th>bar</th>
<th>times_a_week</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Edge</td>
<td>108 Morris Street</td>
<td>Dan</td>
<td>The Edge</td>
<td>1</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 2: Illustration of Semantics of SFW

- NOTE: This is “NOT HOW” the DBMS outputs the result, but “WHAT” is outputs!
  Discard rows that do not satisfy WHERE condition

- SELECT $B$.bar FROM Bar $B$, Frequents $F$
  WHERE $B$.name = $F$.bar AND $F$.drinker LIKE ‘Dan’

<table>
<thead>
<tr>
<th>Bar</th>
<th>address</th>
<th>drinker</th>
<th>bar</th>
<th>times_a_week</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Edge</td>
<td>108 Morris Street</td>
<td>Dan</td>
<td>The Edge</td>
<td>1</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
</tbody>
</table>

Step 3: Illustration of Semantics of SFW

- NOTE: This is “NOT HOW” the DBMS outputs the result, but “WHAT” is outputs!
  Output the “address” output of rows that survived

- SELECT $B$.address FROM Bar $B$, Frequents $F$
  WHERE $B$.name = $F$.bar AND $F$.drinker LIKE ‘Dan’

<table>
<thead>
<tr>
<th>Bar</th>
<th>address</th>
<th>drinker</th>
<th>bar</th>
<th>times_a_week</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Edge</td>
<td>108 Morris Street</td>
<td>Dan</td>
<td>The Edge</td>
<td>1</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
</tbody>
</table>

Final output: Illustration of Semantics of SFW

- NOTE: This is “NOT HOW” the DBMS outputs the result, but “WHAT” is outputs!
  Output the “address” output of rows that survived

- SELECT $B$.address FROM Bar $B$, Frequents $F$
  WHERE $B$.name = $F$.bar AND $F$.drinker LIKE ‘Dan’

<table>
<thead>
<tr>
<th>Bar</th>
<th>address</th>
<th>drinker</th>
<th>bar</th>
<th>times_a_week</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Edge</td>
<td>108 Morris Street</td>
<td>Dan</td>
<td>The Edge</td>
<td>1</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
<tr>
<td>satisfaction</td>
<td>905 W. Main Street</td>
<td>Dan</td>
<td>satisfaction</td>
<td>2</td>
</tr>
</tbody>
</table>