CSE 462 - Databases

Oliver Kennedy
okennedy@buffalo.edu
Why Study Databases?
2 Queries per Second
Interesting Problems

- Algorithms
- Systems
- Hardware
- Theory

Databases
8 of the top 10 Forbes Global 2000 Software & Programming Companies base their business on data management.
What is “Databases”?
Databases

• How do we ask and answer questions about data?

• How do we manipulate and persist data?
Databases

• How do we ask and answer questions about data?
  - accuracy
  - efficiency
  - multiple sources
  - summaries

• How do we manipulate and persist data?
  - consistency
  - correctness
  - parallelism
Database Tools

Techniques:
- Data Modeling
- Cost-Based Optimization

Recipes:
- Join Algorithms
- Index Datastructures

Knowledge:
- The Memory Hierarchy
- Data Consistency
Which tools do you use ...
... and when?
This Course in a Nutshell
This Course in a Nutshell

There might be many correct options…
This Course in a Nutshell

There might be many correct options…
…but some are better than others…
This Course in a Nutshell

There might be many correct options…
…but some are better than others…
…for specific tasks.
This Course in a Nutshell

There might be many correct options…
…but some are better than others…
…for specific tasks.

How do you define ‘correct’ and ‘better’?
This Course in a Nutshell

There might be many correct options…
…but some are better than others…
…for specific tasks.

How do you define ‘correct’ and ‘better’?

How do you find alternatives that are correct?
This Course in a Nutshell

There might be many correct options...  
...but some are better than others...  
...for specific tasks.

How do you define ‘correct’ and ‘better’?

How do you find alternatives that are correct?

How do you find alternatives that are better?
What is ‘Better’?

• **Declarative Queries**: ‘Easy to think about’ vs ‘Fast’

• **Data Layouts**: Space vs Fast Updates vs Fast Queries

• **Parallel Updates**: Reactive vs Proactive Concurrency
Today

- **Logistics**: What you need to know
- **Project Outline**: Build the next big data startup
- **Ways to Fail**: What not to do and why
- **Intro**: So what is a database anyway?
General Course Information
People

- Oliver Kennedy (okennedy@buffalo.edu)
- Jun Chu (jchu6@buffalo.edu)
- Nikhil Londhe (support role only)
Syllabus & Website

http://odin.cse.buffalo.edu/teaching/cse-462

Course Forum: Piazza

Course Project: DµBStep
Course Structure

• **Programming Assignment** (50% of overall grade)
  - 4-Person Groups
  - Build a relational query engine

• **Course Content** (50% of overall grade)
  - **2 Midterm Exams** (5 or 10% of overall grade each)
  - **Comprehensive Final Exam** (20, 25, or 30% of overall grade)
    - Final Grade replaces up to 5% of each midterm’s grade
  - **Homeworks** due on Thursdays (10% of overall grade; drop lowest 2)
Data μBases
Step-by Step
(a.k.a., how to be the next ‘big’ data startup)
Embedded Databases

- SQLite (in your browser, computer, phone, etc…)
- Simple, easy-to-use, declarative data management
- Critical for future tech: Part of Mobile, IoT, Web
Embedded Databases

- SQLite (in your browser, computer, phone, etc…)
- Simple, easy-to-use, declarative data management
- Critical for future tech: Part of Mobile, IoT, Web

Your startup’s goal…

…build (part of) an embedded database
Data µBases
(Step-by-Step)

I give you data (CSV Files + Schema)

I ask you a question about the data (SQL)

You give me an answer
Real World Challenge: You start with…

… an empty GIT repository

… open-source libraries (more on this next week)
Real World Challenge: You get graded on your code’s…

… correctness (do you produce the right answer)
minimum 1/3 of grade for producing the right answer

… speed (how fast did you produce the answer)
+2/3 for meeting/beating the reference implementation
DµBStep

You write code

You push to GIT

DµBStep compiles your code

π-graders run your code

DµBStep emails your group

SUBMIT

JAR

You write code

You push to GIT

DµBStep compiles your code

π-graders run your code

DµBStep emails your group
DµBStep

You write code

You push to GIT

DµBStep compiles your code

π-graders run your code

DµBStep emails your group
Project Outline

A relational query processor

SQL Query ➔ Parser & Translator ➔ Relational Algebra

Optimizer ➔ Statistics

Query Result ➔ Evaluation Engine ➔ Execution Plan
Project Outline

A relational query processor
Project Outline

JSqlParser.jar

SQL Query → Parser & Translator → Relational Algebra → Optimizer → Execution Plan → Evaluation Engine → Query Result

Checkpoint 1

Statistics
Project Outline

JSqlParser.jar

SQL Query ➔ Parser & Translator ➔ Relational Algebra

Checkpoint 2

Optimizer ➔ Statistics

Query Result ➔ Evaluation Engine ➔ Execution Plan
Project Outline

SQL Query → Parser & Translator → Relational Algebra → Optimizer → Execution Plan → Statistics

Query Result → Evaluation Engine → Execution Plan
Projects

- **Checkpoint 0**: “Hello World” Set-up (Due Feb 8)
  - 5% of your overall grade (free points)

- **Checkpoint 1**: Basic SPJU Query Evaluation
  - 15% of your overall grade

- **Checkpoint 2**: “Big” Data & Query Optimization
  - 15% of your overall grade

- **Checkpoint 3**: Pre-computation
  - 15% of your overall grade
Those 5 free points sounded interesting…

… what do I need to do to get them?
Those 5 free points sounded interesting…

… what do I need to do to get them?

http://odin.cse.buffalo.edu/dubstep/checkpoint0.html
5 free points
OMGWTFBBQTooHard
5 free points

• Create a group of up to 4 people.
• Register your group.
• Access your group’s GIT repository.
• Commit a “Hello World” program.
• Hit “Submit”
If it doesn’t work, try again
Submit any project as many times as you need to (before the deadline)

Your grade will not go down if you submit again
Any questions on the project?
Ways to Fail
(do not do these things)

• Start your project at the last minute
• Don’t go to office hours
• Don’t ask questions on Piazza
• Wait until the deadline to submit for the first time
Ways to Fail
(do not do these things)

• Start your project at the last minute
• Don’t go to office hours
• Don’t ask questions on Piazza
• Wait until the deadline to submit for the first time
• Cheat
Academic Integrity

Cheating is submitting any work that you did not perform by yourself as if you did.

References (be sure to cite properly):
Wikipedia, Wikibooks (or similar): **OK**

Public Code:
StackExchange (or similar): **NOT OK**

**Discussing ideas with classmates out of class:**
“A hash index has O(1) lookups”: **OK** (except during exams 😇)

**Sharing code or answers with classmates:**
“Just have a look at how I implemented it”: **NOT OK**
MOSS

Submission Overlap (Ignoring Library Code)
MOSS-Details

Identical Code Structure

Code in Case Statement

Code in “Operator Class”
Academic Integrity

Zero Tolerance: If I catch you submitting someone else’s code, you will fail the class.

Group Responsibility: If your teammate cheats on a group project, the entire group will be penalized.

Share Code, Share Blame: If someone else submits your code as their own, you will be penalized as well.
Questions/Concerns?
What does a data-management system do?
Data Management

**Analysis:** Answering user-provided questions about a dataset

What kind of tools can we give end-users?

- Declarative Languages
- Organizational Datastructures (e.g., Indexes)

**Manipulation:** Safely persisting and sharing data updates

What kind of tools can we give end-users?

- Consistency Primitives
- Data Validation Primitives
Data

vs

45
Data

vs

`
{
  "firstName": "John",
  "lastName": "Smith",
  "age": 25,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": 10021
  },
  "phoneNumbers": [
    {
      "type": "home",
      "number": "212 555-1234"
    },
    {
      "type": "fax",
      "number": "646 555-4567"
    }
  ]
}
`
Data

Databases exploit the data’s structure!
So let’s talk structure...
Types
Types

Integer

Floating Point Number

String

List/Array

Bag

Set

Struct

Dictionary/Object
Types

**Primitive**
- Integer
- Floating Point Number

**Collection**
- List/Array
- Bag
- Set

**Tuple**
- Dictionary/Object
- Struct
- String
Type Glossary

- **Primitive**: Basic building blocks like Int, Float, Char, *String*

- **Tuple**: Several ‘fields’ of different types. (N-Tuple = N fields)
  - A Tuple has a ‘schema’ defining names/types for each field

- **Set**: A collection of unique records, all of the same type

- **Bag**: An unordered collection of records, all of the same type

- **List**: An ordered collection of records, all of the same type
Relational Database Glossary
Relational Database Glossary
**Relational Database Glossary**

- **Schema**: Specifies the name of the relation, plus the name and type of each column.

**The Data**

| James, Kirk, 2260 |
| Jean Luc, Picard, 2360 |
| Benjamin, Sisko, 2365 |

**Officers**

```
Officers(
    firstname: string,
    lastname: string,
    id: int
)
```
Relational Database Glossary

- **Relation**
- **Schema**
- **Instance**

Columns

\((# = \text{degree/arity})\)

Specifies the name of the relation, plus the name and type of each column

**Officers**

\[
\text{ Officers}(
  \text{ firstname: string, }
  \text{ lastname: string, }
  \text{ id: int }
)\]

The Data

- [James, Kirk, 2260]
- [Jean Luc, Picard, 2360]
- [Benjamin, Sisko, 2365]
Relational Database Glossary

The Data

[James, Kirk, 2260]
[Jean Luc, Picard, 2360]
[Benjamin, Sisko, 2365]

Columns
(# = degree/arity)

Specifies the name of the relation, plus the name and type of each column

Officers(
  firstname: string,
  lastname: string,
  id: int
)

Rows
(# = cardinality)

Relational Database

Schema

Instance

Relation

The Data

Officer(William, Riker, 1106)
A relation is a set of tuples (rows) with the same schema.
Why?

Your data is currently an *Unordered Set of 100-attribute Tuples*

Tomorrow, you’ll be repeatedly asked for 1 specific attribute of 5 specific rows identified by the first attribute

Can you do better?
Better Idea: Rewrite data into a 99-Tuple of Maps keyed on the 1st attribute

This representation is equivalent, and better for your needs.

Declarative specs make it easier to find equivalences.