CS564: Database Management Systems

Lecture 1: Course Overview

Acks: Chris Ré
data is the new oil
Big science is data driven.
Increasingly many companies see themselves as data driven.
Even more “traditional” companies...

DIGITAL INDUSTRIAL COMPANY

Bloomberg Businessweek: How GE Became A 124-Year-Old Startup

The story tracks GE’s digital transformation from its inception after the financial crisis in 2008. It started with a broad idea. “I said, ‘Look, we need to start building analytic capability, big data capability, and let’s do it in California,’” Immelt told the magazine. “That was as sophisticated as my original thinking was.”
The world is increasingly driven by data...

This class teaches the basics of how to use & manage data.
Today’s Lecture

1. Introduction, admin & setup
   • ACTIVITY: Jupyter “Hello World!”

2. Overview of the relational data model
   • ACTIVITY: SQL in Jupyter

3. Overview of DBMS topics: Key concepts & challenges
1. Introduction, admin & setup
What you will learn about in this section

1. Motivation for studying DBs
2. Administrative structure
3. Course logistics
4. Overview of lecture coverage
5. ACTIVITY: Jupyter “Hello World!”
New tech. Same Principles.
Why should you study databases?

• **Mercenary-make more $$$:**
  • Startups need DB talent right away = low employee #
  • Massive industry...

• **Intellectual:**
  • Science: data poor to data rich
    • No idea how to handle the data!
  • Fundamental ideas to/from all of CS:
    • Systems, theory, AI, logic, stats, analysis....

Many great computer systems ideas started in DB.
What this course is (and is not)

• Discuss **fundamentals of data management**
  • How to design databases, query databases, build applications with them.
  • How to debug them when they go wrong!
  • **Not** how to be a DBA or how to tune Oracle 12g.

• We’ll cover **how database management systems work**

• And some **basic principles of how to build** them
Who we are...

Instructor (me) Theo Rekatsinas

• Faculty in the Computer Sciences and part of the UW-Database Group
• Research: data integration and cleaning, statistical analytics, and machine learning.
• thodrek@cs.wisc.edu
• Office hours: Wed 4:00-5:00pm (after class), Fri 11:00am-12:00 pm @CS 4361
Teaching Assistants (TAs)
“TAs are humans too!”
Teaching Assistants (TAs)
“TAs are humans too!”
Section 1 > Administrative > Course Staff

Minzhen

Vishnu
Communication w/ Course Staff

• Piazza https://piazza.com/wisc/fall2017/cs5643

• Class email: cs564fall17@gmail.com

• Office hours

• By appointment!

The goal is to get you to answer each other’s questions so you can benefit and learn from each other.

OHs are listed on the course website!
Course Website:

https://thodrek.github.io/cs564-fall17/

Course Email:

cs564fall17@gmail.com
Lectures

• Lecture slides cover essential material
  • This is your best reference.
  • We are trying to get away from book, but we will have pointers
  • Recommended textbooks listed on website

• Try to cover same thing in many ways: Lecture, lecture notes, homework, exams (no shock)
  • Attendance makes your life easier...
Attendance

• You should attend lectures plus guest lecture
  • Guest lectures are fun. Great guests: they want to meet you! Show up!

• Attendance is for your benefit...
  • People who did not attend did worse 😞
  • People who did not attend used more course resources 😞
  • People who did not attend were less happy with the course 😞
Graded Elements

• **Two** Problem Sets (15%)

• Programming project (30%)
  • Split into **four parts**
  • Auction base: Experience with a DB application.
  • Implementation of DB internals

• Midterm (20%)

• Final exam (35%)

All due dates are posted on website!!!
Un-Graded Elements

• Readings provided to help you!
  • Only items in lecture, homework, or project are fair game.

• Activities are again mainly to help / be fun!
  • Will occur during class- not graded, but count as part of lecture material (fair game as well)

• Jupyter Notebooks provided
  • These are optional but hopefully helpful.
  • Redesigned so that you can ‘interactively replay’ parts of lecture
What is expected from you

• **Attend lectures**
  • If you don’t, it’s *at your own peril*

• **Be active and think critically**
  • Ask questions, post comments on forums

• **Do programming and homework projects**
  • Start early and *be honest*

• **Study for tests and exams**
Problem Sets

• Two problems sets at the beginning

• Individual assignments
  • Python plus Jupiter notebooks

• 1 week per problem set
  • Ask questions, post comments on forums
  • Start early!
Project

- **Split into four parts**
  - Two parts cover DB applications
  - Two parts cover DB internals

- **In groups of 3**
  - One person per team emails the group info by Wednesday 9/13.
  - Use [cs564fall17@gmail.com](mailto:cs564fall17@gmail.com) subject should be **CS564-3: Project Group**
  - Write your names and University ID

- **Python (DB design) and C++ (DB internals)**

- **Varying duration for different parts (2 weeks at least)**
  - Exact dates posted on website
  - Ask questions, post comments on forums
  - Start early!
To encourage awesomeness

Bonus assignments, activities, and projects... Some extremes...

1. I was hung over when I took the test. Intended to make up for silly mistakes.

2. I want to be a research star! There will be some challenging assignments that could indicate possible publication (e.g., ACM SIGMOD undergrad competition)
Lectures: 1\textsuperscript{st} part - from a user’s perspective

1. **Foundations**: Relational data models & SQL
   - Lectures 2-4
   - How to manipulate data with SQL, a declarative language
     - reduced expressive power but the system can do more for you

2. **Database Design**: Design theory and constraints
   - Lectures 5-8
   - Designing relational schema to keep your data from getting corrupted
Lectures: 2\textsuperscript{nd} part – database internals

3. Introduction to database systems
   - Lectures 9-11
   - Data Storage and IO models
   - Buffer Manager and File Organization
   - External sorting

4. Indexing and Hashing
   - Lectures 12-15
   - Intro to indexing
   - B+ Tree, Hash, and Bitmap Indexes

5. Query processing
   - Lectures 16-20
   - Access methods and operators
   - Joins
   - Relational algebra and Query optimization
Lectures: 3rd part – transactions

6. Transactions
   - Lectures 21-22
   - Transactions from a user’s perspective
   - Logging and Locking

7. Bonus
   - Guest Lecture and Lecture 23
   - Stratis Viglas from Google
   - Machine Learning meets Data Management
Lectures: A note about format of notes

*These are asides / notes (still need to know these in general!)*

Definitions in blue with **concept being defined** bold & underlined

*Main point of slide / key takeaway at bottom*

*Warnings- pay attention here!*

*Take note!!*
Jupyter Notebook “Hello World”

- Jupyter notebooks are interactive shells which **save output in a nice notebook format**
  - They also can display markdown, LaTeX, HTML, js...

- You’ll use these for
  - in-class activities
  - interactive lecture supplements/recaps
  - homeworks, projects, etc.- if helpful!

FYI: “Jupyter Notebook” are also called iPython notebooks but they handle other languages too.

Note: you **do need to know or learn python** for this course!
Jupyter Notebook Setup

1. **HIGHLY RECOMMENDED.** Install **on your laptop** via the instructions on the next slide / Piazza

2. Other options running via one of the alternative methods:
   1. Ubuntu VM.
   2. CS Machines.

3. Come to office hours if you need help with installation!

---

As a general policy in upper-level CS courses, **Windows is not officially supported.**
Jupyter Notebook Setup

https://thodrek.github.io/cs564-fall17/misc/jupyter_install.html
Activity-1-1.ipynb
2. Overview of the relational data model
What you will learn about in this section

1. Definition of DBMS

2. Data models & the relational data model

3. Schemas & data independence

4. ACTIVITY: Jupyter + SQL
What is a DBMS?

• A large, integrated collection of data

• Models a real-world *enterprise*
  • *Entities* (e.g., Students, Courses)
  • *Relationships* (e.g., Alice is enrolled in CS564)

A **Database Management System (DBMS)** is a piece of software designed to store and manage databases
A Motivating, Running Example

• Consider building a course management system (CMS):

  • Students
  • Courses
  • Professors

  \{ Entities \}

  • Who takes what
  • Who teaches what

  \{ Relationships \}
Data models

• A **data model** is a collection of concepts for describing data

  • The relational model of data is the most widely used model today
    • Main Concept: the *relation*--essentially, a table

• A **schema** is a description of a particular collection of data, **using the given data model**

  • E.g. every *relation* in a relational data model has a *schema* describing types, etc.
Modeling the Course Management System

• **Logical Schema**
  • Students(sid: string, name: string, gpa: float)
  • Courses(cid: string, cname: string, credits: int)
  • Enrolled(sid: string, cid: string, grade: string)

<table>
<thead>
<tr>
<th>sid</th>
<th>Name</th>
<th>Gpa</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>Bob</td>
<td>3.2</td>
</tr>
<tr>
<td>123</td>
<td>Mary</td>
<td>3.8</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>cid</th>
<th>cname</th>
<th>credits</th>
</tr>
</thead>
<tbody>
<tr>
<td>564</td>
<td>564-2</td>
<td>4</td>
</tr>
<tr>
<td>308</td>
<td>417</td>
<td>2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>sid</th>
<th>cid</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>564</td>
<td>A</td>
</tr>
</tbody>
</table>
Modeling the Course Management System

• **Logical Schema**
  - Students(sid: string, name: string, gpa: float)
  - Courses(cid: string, cname: string, credits: int)
  - Enrolled(sid: string, cid: string, grade: string)
Other Schemata...

• **Physical Schema**: describes data layout
  • Relations as unordered files
  • Some data in sorted order (index)

• **Logical Schema**: Previous slide

• **External Schema**: (Views)
  • Course_info(cid: string, enrollment: integer)
  • Derived from other tables
Data independence

Concept: Applications do not need to worry about *how the data is structured and stored*

**Logical data independence:** protection from changes in the *logical structure of the data*

I.e. should not need to ask: can we add a new entity or attribute without rewriting the application?

**Physical data independence:** protection from *physical layout changes*

I.e. should not need to ask: which disks are the data stored on? Is the data indexed?

One of the most important reasons to use a DBMS
Activity-1-2.ipynb
3. Overview of DBMS topics

Key concepts & challenges
What you will learn about in this section

1. Transactions

2. Concurrency & locking

3. Atomicity & logging

4. Summary
Challenges with Many Users

• Suppose that our CMS application serves 1000’s of users or more—what are some challenges?

  • Security: Different users, different roles
  
  • Performance: Need to provide concurrent access
  
  • Consistency: Concurrency can lead to update problems

We won’t look at too much in this course, but is extremely important

Disk/SSD access is slow, DBMS hide the latency by doing more CPU work concurrently

DBMS allows user to write programs as if they were the only user
Transactions

• A key concept is the **transaction (TXN)**: an **atomic** sequence of db actions (reads/writes)

<table>
<thead>
<tr>
<th>Acct</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>a10</td>
<td>20,000</td>
</tr>
<tr>
<td>a20</td>
<td>15,000</td>
</tr>
</tbody>
</table>

Transfer $3k from a10 to a20:
1. Debit $3k from a10
2. Credit $3k to a20

Written naively, in which states is **atomicity** preserved?

• Crash before 1,
• After 1 but before 2,
• After 2.

**Atomicity**: An action either completes **entirely or not at all**

<table>
<thead>
<tr>
<th>Acct</th>
<th>Balance</th>
</tr>
</thead>
<tbody>
<tr>
<td>a10</td>
<td>17,000</td>
</tr>
<tr>
<td>a20</td>
<td>18,000</td>
</tr>
</tbody>
</table>

**DB Always preserves atomicity!**
Transactions

• A key concept is the transaction (TXN): an atomic sequence of db actions (reads/writes)
  • If a user cancels a TXN, it should be as if nothing happened!

• Transactions leave the DB in a consistent state
  • Users may write integrity constraints, e.g., ‘each course is assigned to exactly one room’

However, note that the DBMS does not understand the real meaning of the constraints—consistency burden is still on the user!

Atomicity: An action either completes entirely or not at all

Consistency: An action results in a state which conforms to all integrity constraints
Challenge: Scheduling Concurrent Transactions

• The DBMS ensures that the execution of \( \{T_1, \ldots, T_n\} \) is equivalent to some *serial* execution

• One way to accomplish this: **Locking**
  • Before reading or writing, transaction requires a lock from DBMS, holds until the end

• **Key Idea**: If \( T_i \) wants to write to an item \( x \) and \( T_j \) wants to read \( x \), then \( T_i, T_j \) conflict. Solution via locking:
  • only one winner gets the lock
  • loser is blocked (waits) until winner finishes

A set of TXNs is **isolated** if their effect is as if all were executed serially

What if \( T_i \) and \( T_j \) need \( X \) and \( Y \), and \( T_i \) asks for \( X \) before \( T_j \), and \( T_j \) asks for \( Y \) before \( T_i \)?

-> **Deadlock!** One is aborted...

All concurrency issues handled by the DBMS...
Ensuring Atomicity & Durability

• DBMS ensures **atomicity** even if a TXN crashes!

• One way to accomplish this: **Write-ahead logging (WAL)**

• **Key Idea:** Keep a log of all the writes done.
  • After a crash, the partially executed TXNs are undone using the log

**Write-ahead Logging (WAL):** Before any action is finalized, a corresponding log entry is forced to disk

*We assume that the log is on “stable” storage*

All atomicity issues also handled by the DBMS...
A Well-Designed DBMS makes many people happy!

- End users and DBMS vendors
  - Reduces cost and makes money

- DB application programmers
  - Can handle more users, faster, for cheaper, and with better reliability / security guarantees!

- Database administrators (DBA)
  - Easier time of designing logical/physical schema, handling security/authorization, tuning, crash recovery, and more…

Must still understand DB internals
Summary of DBMS

• DBMS are used to maintain, query, and manage large datasets.
  • Provide concurrency, recovery from crashes, quick application development, integrity, and security

• Key abstractions give data independence

• DBMS R&D is one of the broadest, most exciting fields in CS. Fact!