

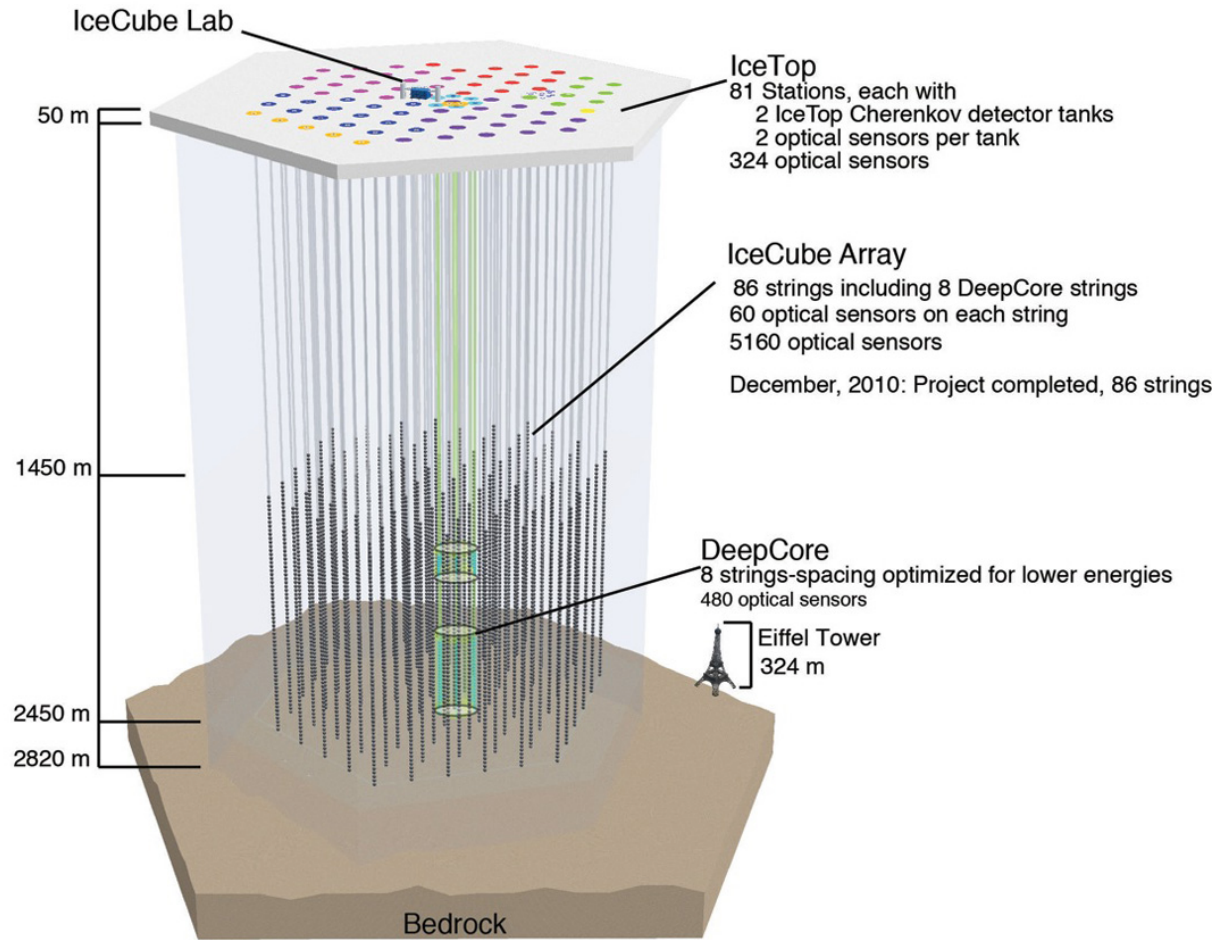
CS145: Intro to Database Management Systems

Lecture 1: Course Overview

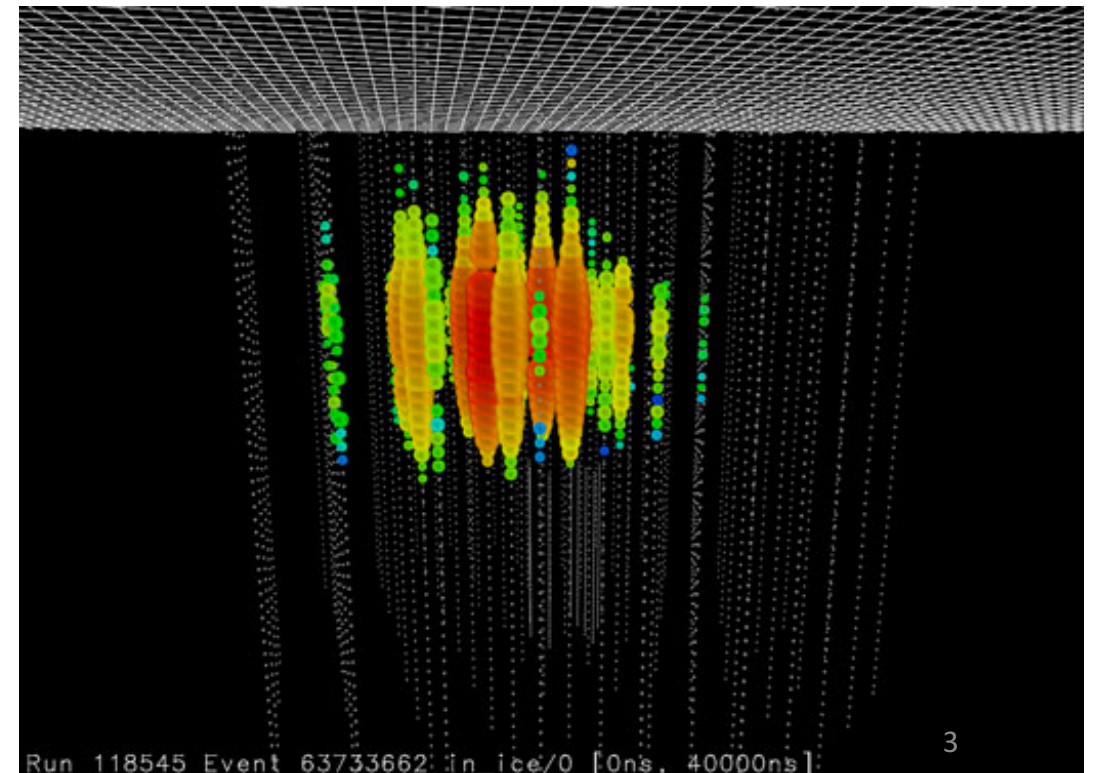
“Data is the Future”

- My cab driver in Pittsburg

IceCube Neutrino Observatory.



Big science is data driven.





All of society is online.



Data analysis in the fight
against human trafficking.

*New York DA use MEMEX
Data for all trafficking
investigations this year.*



Increasingly many companies see themselves as **data driven**.

Even more “traditional” companies...



https://www.youtube.com/watch?v=iH9C_ylGOq4

I wanted to show you “Smart Pipe from Comedy Central’s Infomercial” but Alex said it’s a bad idea (it’s also 11 minutes long)

The world is increasingly
driven by data...

This class teaches the basics of
how to use & manage data.

This year: Turing Award for Innovations in RDBMSs

“The Nobel of Computing”

- 2014 A.M. Turing Award Winner: Michael Stonebraker
- Helped to invent many RDBMS (Relational DBMS) concepts:
 - Query modification
 - The Object-Relational model
 - More recently: work on column-store, streaming data
- Made / helped to start many popular RDBMS implementations:
 - Postgres, Vertica, Streambase, VoltDB, ...



The relational data model is one of the most important concepts in computing!

Today's Lecture

1. Introduction, admin & setup
 - ACTIVITY: IPython “Hello World!”
2. Overview of the relational data model
 - ACTIVITY: SQL in IPython
3. Overview of DBMS topics: Key concepts & challenges

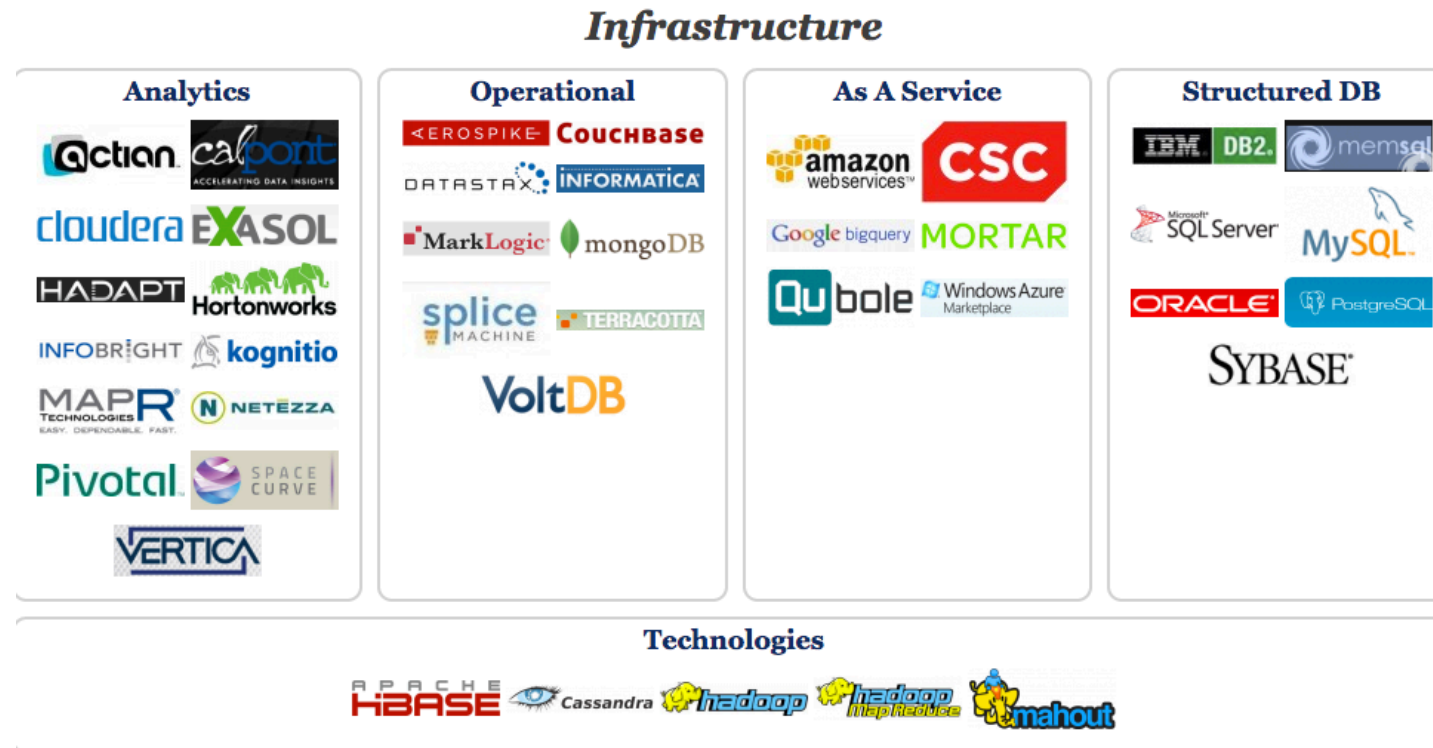
1. Introduction, admin & setup

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: IPython “Hello World!”

Big Data Landscape...

Infrastructure is Changing



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**
- But not **the principles of how to build them** 😞
 - see 245, 345, and 346.

Who we are...

Instructor (me) Chris Ré

Sounds like "Ray"

- Faculty in the InfoLab
- Research: theory of data processing, statistical analytics, and machine reading.
- chrismre@cs.stanford.edu
- Office hours: T/Th 4:30-5:30, Gates 433



Course Assistants (CAs)

“Remember, CAs are people too!”

- Probably some CA



Alex



Ari



Yuchen



Stephanie



Arun



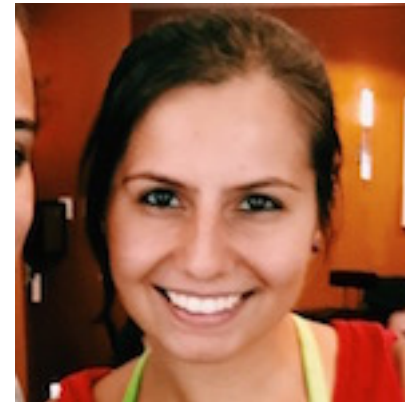
Vishnu



Shubham



Kevin



Cagla



Jed

Communication w/ Course Staff

- Piazza
- Office hours
- *By appointment!*

*OHs are (or will soon be)
listed on the course website!*

Piazza



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

If I troll you on Piazza, take it as a sign of *love*.

Important!

Students with documented disabilities should send in their accommodation letter from O.A.E. (Office of Accessible Education) by the **end of this week** to *Alex Ratner (Head TA) & cc' me*.

Course Website:

cs145.stanford.edu

Lectures

- Lecture slides cover **essential material**
 - This is your best reference.
 - We are trying to get away from book, but do have pointers
- Try to cover same thing in **many ways**: Lecture, lecture notes, homework, exams (no shock)
 - Attendance makes your life easier...
 - 8 lectures + all guest lectures are mandatory!

Graded Elements

- Attendance (10%)
- Problem Sets (20%)
- Programming project (20%)
 - Auction base. Up now!
 - Experience with a database application.
- Midterm (20%)
- Final exam (30%)

All but the final assignment are due on Tuesday before class.

*For SCPD students only:
Attendance will not be a component of grading;
distribution will be scaled amongst the rest proportionately*

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)
- IPython 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**

Going beyond the requirements...

SIGMOD15

<http://www.sigmod.org/sigmod-awards/sigmod-awards#undergraduate>

Undergrad Research Award Winners.

Adam Perelman *Dunce Cap: Compiling Worst-Case Optimal Query Plans and Beyond*

Former CS 145'ers!

Susan Tu *Dunce Cap: Query Plans Using Generalized Hypertree Decompositions*



EMPTYHEADED

SIGMOD15 was in Australia



To encourage more awesomeness

Github for course material & bonus projects... Some extremes...

1. **I was hung over when I took the test.** *Intended to make up for silly mistakes.*
2. **I want to make it easier for future generations.** Visualizations & improvements to advanced topics.
3. ***I want to be a research star!*** These are challenging assignments that could indicate possible publication.

If your pull request is approved, TA will give you bonus points.

Optional Elements

- Tutorials for github on course page.
- Please use Piazza for questions on the topics (not github).
- These are **optional** elements of the course.

Lectures: 1st half - from a user's perspective

1. **Foundations:** Relational data models & SQL

- Lectures 2-3
- 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 4-5, 7
- Designing relational schema to keep your data from getting corrupted

3. **Transactions:** Syntax & supporting systems

- Lectures 8-9
- A programmer's abstraction for data consistency

Lectures: 2nd half - understanding how it works

4. Introduction to database systems

- Lectures 12-16
- Indexing
- External Memory Algorithms (IO model) for sorting, joins, etc.
- Basics of query optimization (Cost Estimates)
- Relational algebra

5. Specialized and New Data Processing Systems

- Lectures 17-19
- Key-Value Stores
- Hadoop and its 10 year anniversary
- SparkSQL. The re-rise of SQL
- “Dark data” systems & current intersections with ML & AI

Lectures: A note about format of notes

Take note!!

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!

IPython Notebook “Hello World”

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

FYI: IPython Notebook is now called “Jupyter Notebook” and handles other languages too. Same thing basically.

```
In [9]: display(i)
```

IP[y]: IPython
Interactive Computing

```
In [3]: from IPython.display import SVG  
        SVG(filename='python-logo.svg')
```

Out[3]:

The image shows a screenshot of an IPython notebook interface. It displays two code cells. The first cell contains the command 'display(i)' and the output is the IPython logo, which consists of the letters 'IP' in blue and 'y' in yellow, followed by the text 'IPython Interactive Computing'. The second cell contains the code 'from IPython.display import SVG' and 'SVG(filename='python-logo.svg')', and the output is the Python logo, which is a blue and yellow snake logo followed by the word 'python' in a grey font with a trademark symbol.

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

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

Python Notebook Setup

1. First, try installing **on your laptop** via the instructions on the next slide / Piazza
2. Try running via one of the alternative methods:
 1. **Ubuntu VM** (coming soon- see Piazza!)
 2. **Corn** via X11 *OR* port forwarding (see instructions on Piazza)
3. Come to our **Installation Office Hours** later this week (TBA on Piazza)!

Please help out your peers by posting issues / solutions on Piazza!

As a general policy in upper-level CS courses, **Windows is not officially supported**. However we are making a best-effort attempt to provide some solutions here!

IPython Notebook Setup

On your laptop: *Preferred method!!*

1. *Make sure Python, pip & git installed*
2. *Open a terminal and do:*

```
> git clone https://github.com/HazyResearch/cs145-
notebooks.git
> cd cs145-notebooks
> pip install --user --upgrade "ipython[notebook]"
jupyter ipython-sql
> ipython notebook
```

On corn.stanford.edu:

1. *Make sure X11 is installed on your laptop*
2. *Open a terminal and do:*

```
> ssh -Y <your-sunet-id>@corn
> git clone https://github.com/HazyResearch/cs145-
notebooks.git
> cd cs145-notebooks
> pip install --user --upgrade "ipython[notebook]"
jupyter ipython-sql
> ipython notebook
```

CAs will be coming around to help with setup & installation

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: IPython + 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 145)

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

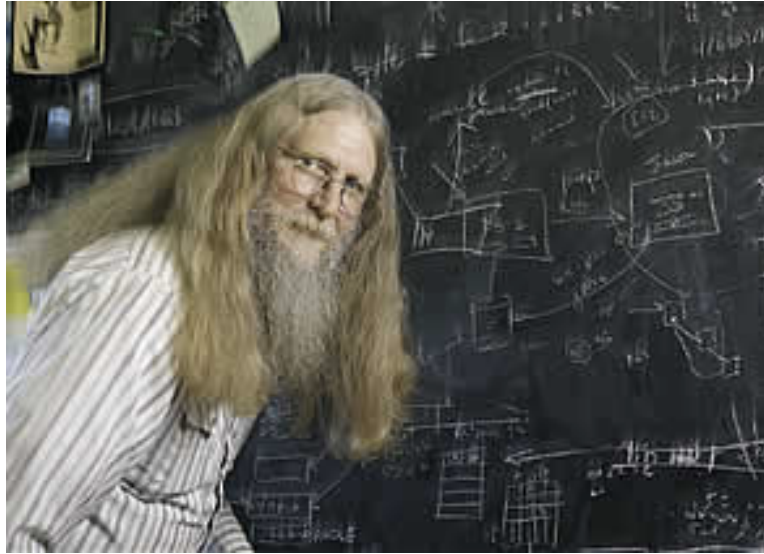
} *Entities*

- 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.



“Relational databases form the bedrock of western civilization”

- Bruce Lindsay, IBM Research

Modeling the CMS

- *Logical Schema*

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

sid	Name	Gpa
101	Bob	3.2
123	Mary	3.8

Students

Relations

sid	cid	Grade
123	564	A

Enrolled

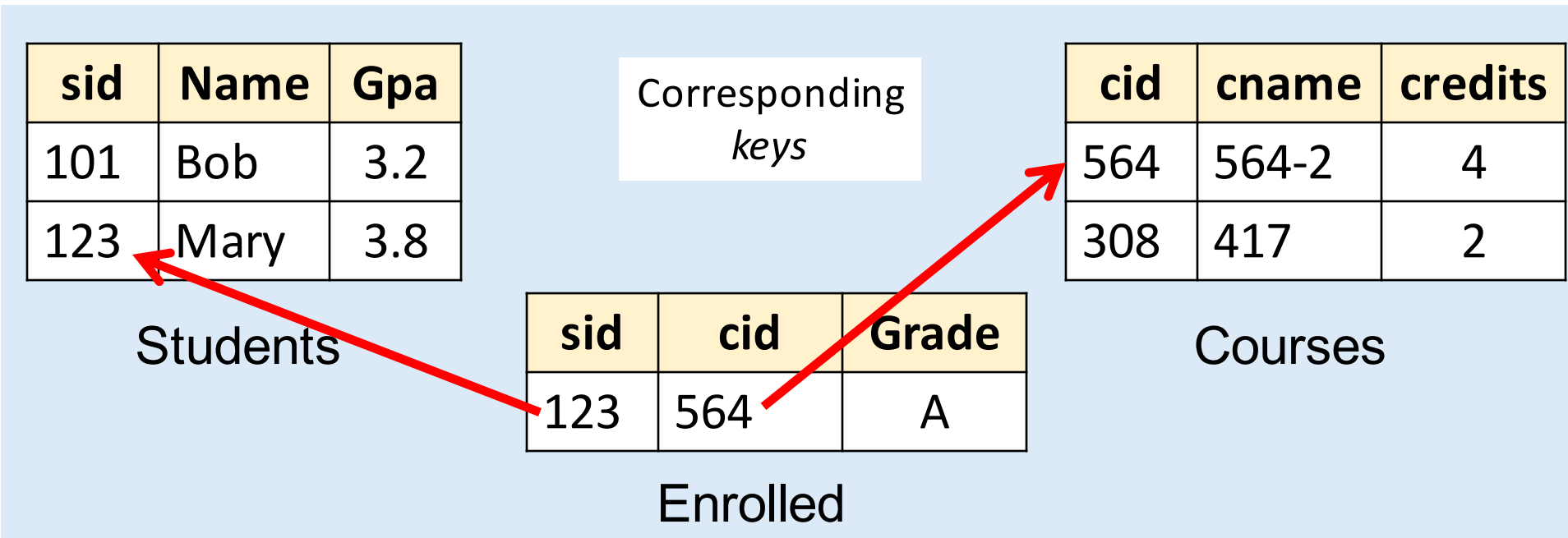
cid	cname	credits
564	564-2	4
308	417	2

Courses

Modeling the CMS

- *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)



Administrators

- *Logical Schema*: Previous slide

- *External Schema*: (Views)
 - `Course_info(cid: string, enrollment: integer)`
 - Derived from other tables



Applications

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

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

- Performance: Need to provide concurrent access

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

- Consistency: Concurrency can lead to update problems

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)
 - 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'

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

Consistency: An action results in a state which conforms to all integrity constraints

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

Scheduling Concurrent Transactions

- The DBMS ensures that the execution of $\{T_1, \dots, T_n\}$ 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!**