Special Topics:
Self-Driving Database Management Systems

Database Automation
Background

@Andy_Pavlo // 15-799 // Spring 2022
TODAY’S AGENDA

Course Logistics Overview
History of Database Automation
WHY YOU SHOULD TAKE THIS COURSE

There are more databases than ever.
Everybody has database problems.
Humans are not scalable.
Automation is the future.

Research: The problem is hard / interesting.
Industry: Every database company needs this.
COURSE OBJECTIVES

Learn about state-of-the-art methods for automated database optimization and management.

Students will become proficient in:
→ Writing database automation code
→ Proper documentation + testing
→ Code reviews
→ Working on a large code base
We will focus on how to optimize single-node relational database management systems via existing interfaces (e.g., SQL).

→ Non-relational DBMSs (NoSQL) also have these problems, but Andy is an apostle of Stonebraker.

We will ignore distributed deployment problems and ML-derived internal components ("learned").
COURSE TOPICS

Indexes
Knobs / Parameter Configuration
Partitioning
Query Optimization
Workload Modeling
System Behavior Modeling
Autonomous Database Systems
BACKGROUND

I assume that you have already taken an intro course on databases (e.g., 15-445/645).
→ Things that we will not cover:
  SQL, Serializability Theory, Relational Algebra, Basic Algorithms + Data Structures.

This is also not a ML course. We will not cover ML algorithms beyond what is discussed in papers.
→ Andy only cares about databases and whatever he can use to make databases run better.
COURSE LOGISTICS

Course Policies + Schedule:
→ Refer to course web page.

Academic Honesty:
→ Refer to CMU policy page.
→ If you’re not sure, ask me.
OFFICE HOURS

Fridays @ 2:30pm ET over Zoom

Things that we can talk about:
→ Issues on implementing projects
→ Paper clarifications/discussion.
→ How to get a database dev job.
→ Life after the pandemic.
TEACHING ASSISTANTS

Head TA: Matt Butrovich
→ 4th Year PhD Student (CSD)
→ Reformed Gang Member (LAX)
→ #1 Ranked CMU-DB PhD Student
→ Recently married and ready to put his felonious past behind him.
COURSE RUBRIC

Reading Assignments
Paper Presentations
Programming Projects
One mandatory reading per class (👑). You can skip three readings during the semester.

You must submit a synopsis before class:
→ Overview of the main idea (three sentences).
→ Three strengths of method (one sentence each).
→ Three weaknesses of method (one sentence each).
→ Workloads evaluated (one sentence).

Submission Form:
https://cmudb.io/15799-s22-submit
Each review must be your own writing.

You may **not** copy text from the papers or other sources that you find on the web.

Plagiarism will **not** be tolerated.
See CMU's Policy on Academic Integrity for additional information.
Starting January 26th, one student will be assigned to present the assigned reading for that day and lead the discussion.

→ There are not an even number of papers to present per student, so we can double up.
→ You are allowed / encouraged to reach out to authors and use their slides as the starting point.

The rest of the class should participate in the discussion during and after presentation.

I will send out the sign-up sheet later this week.
PROGRAMMING PROJECTS

All projects will be implemented using PostgreSQL and NoisePage (pilot not DBMS).

You should do all development and testing on your local machine.

You will do all benchmarking using self-managed databases on Amazon EC2 (don't use RDS)

→ We will provide details next week and credits.
→ If you blow out your credit card, I cannot help you.
You will build an automatic tuning tool for PostgreSQL. It is purposely designed to be open ended to let to play around.

→ Anything is on the table as long as the DBMS doesn't crash, lose data, or produce incorrect query results.

We will provide you with infrastructure to run workloads with BenchBase.

Project #1 will be completed individually.
PROJECT #2

Each group (3-4 people) will choose a project that satisfies the following criteria:
→ Relevant to the materials discussed in class.
→ Requires a significant programming effort from all team members.
→ Unique (i.e., two groups cannot pick same idea).
→ Approved by me.

You don’t have to pick a topic until after you come back from Spring Break.
We will provide sample project topics.
PROJECT #3

The class will work together to build a working end-to-end autonomous database prototype.

We will spend a portion of each class time after Spring Break to discuss development issues.
These projects must be all your own code.

You may **not** copy source code from other groups or the web.

Plagiarism will **not** be tolerated. See [CMU's Policy on Academic Integrity](#) for additional information.
GRADE BREAKDOWN

Reading Reviews (20%)
Presentations + Participation (20%)
Project #1 (20%)
Project #2 (25%)
Project #3 (15%)
COURSE SLACK CHANNEL

Andy will send invitations to everyone enrolled to the CMU-DB Slack channel for the course.

If you have a technical question about the projects, please use Slack.
→ Don’t email me or TAs directly.

All non-project questions should be sent to me.
Vaccination Database Tech Talks
→ Mondays @ 4:30pm (starting on Jan 31st)
→ https://db.cs.cmu.edu/seminar2022-booster
Database Automation
The proliferation of the Internet means that it is easy to create an application that can have thousands of users in a short amount of time.

Instead of managing data files on its own, the application should use a DBMS that is responsible for storing and querying the database.
Physical data independence allows a DBMS to change many aspects of a database without having to change application code.

Example: If a query's execution time is too slow because it performs a sequential scan, one can add a new index to make it run faster without having to change the query's SQL.
DATABASE OPTIMIZATION

Physical Database Design
Knob Configuration
Query Tuning
Hardware Provisioning
PHYSICAL DATABASE DESIGN

Storage
→ Organization (row vs. columnar), Compression, Encoding, Denormalization, Clustering.

Indexes
→ Keys, Data Structures, Include Columns, Partial Indexes, Expression Indexes.

Partitioning
→ Sharding Keys (Horizontal), Column Splits (Vertical), Schemes, Replication.

Result Caching
→ Materialized Views, Memoization, Zone Maps.
KNOB CONFIGURATION

Resource Knobs
→ How much hardware resources to use for an internal component or task.

Policy Knobs
→ The execution behavior of the DBMS for some task.

Location Knobs
→ Filepaths, directories, network info (host/port).
Plan Hints
→ Force the DBMS to execute a query in a specific manner.
→ Join ordering, access methods, execution strategies

Rewriting
→ Changing SQL syntax to improve the execution behavior of the DBMS.

Rerouting
→ Redirect read-only queries to replicas with idle resources.

Optimizer Statistics
→ Make sure that the DBMS has the freshest stats about the database as possible so that it makes "good" choices.
HARDWARE PROVISIONING

Vertical Scaling
-> Change the hardware capabilities of a single node (CPU, memory, disk, network).

Horizontal Scaling
-> Change the number of physical nodes in a logical database.
-> This for either distributed DBMS or replicated DBMS.
Traditionally, it has been the role of the database administrator to manage all aspects of the DBMS.

With so many different options for optimizing each a DBMS, DBAs are typically an expert in one or DBMS type.

→ Core concepts are universal, but implementation specifics and behavior of all DBMSs are different.
What is the mean salary of a DBA in the US?

→ Average DBA Salary (2020): $101,090
→ Personnel is estimated to be ~50% of the total-cost-of-ownership (TCO) of a DBMS.

The scale and complexity of DBMS installations have surpassed the ability of humans to effectively manage and optimize them individually.

Source: https://www.bls.gov/oes/current/oes151141.htm
Source: https://www.highbeam.com/doc/1P3-1149052351.html
The first work on automated database optimization tools started in the 1970s.

These were tools that made recommendations to human operators on what changes to make to a database.
AUTOMATED DATABASE TUNING

1970s: Physical Design Tuners
1980s: Physical Design w/ Traces
1990s: Partitioning + Placement
2000s: Microsoft AutoAdmin
2010s: ML-enhanced Methods
2020s: Self-Driving Systems
1970s: Physical Design Tuners
1980s: Physical Design w/ Traces
1990s: Partitioning + Placement
2000s: Microsoft AutoAdmin
2010s: ML-enhanced Methods
2020s: Self-Driving Systems
TARGET OBJECTIVE

An automated system must be instructed by a human on what objective it should optimize.
→ Throughput
→ Latency
→ Cost
→ Hardware Utilization

A system could use multiple objectives where some have different priorities or provided as a constraint.
→ Example: Minimize cost but maintain latency threshold.
A change to the database is called an action. → Build index, change knob, etc.

An automated tool decides what actions to apply to the database to improve some target objective.

The number of possible actions available to optimize a DBMS is clearly NP-Complete. → Just picking indexes for a database is NP-Complete.
COST MODELS

It is too expensive to try out all possible design decisions in a database to find options.
→ Copying data is expensive / time-consuming.
→ Some changes also do not take effect until restarting.
→ Actions are not independent.

We will see different methods to estimate the cost/benefit of actions through a combination of external and internal cost models.
IMPORTANT TUNING RULES

Do **not** require changes to the application.
Do **not** lose data.
Do **not** degrade observable correctness.
Do **not** incur unscheduled unavailability.
Do **not** cause observable unexpected behavior.

In general, we must avoid any decision that requires a human to make a **value judgement**.
PARTING THOUGHTS

The previous 50 years of research focused on the problem of **what** to tune in a DBMS.

This course will also cover the **when** and **how** problems that are necessary to achieve parity with human experts.
SELF-DRIVING DATABASES OVERVIEW

Make sure that you submit the first reading review

https://cmudb.io/15799-s22-submit