50.043 Database and Big Data Systems

Introduction

Roy Ka-Wei Lee
Assistant Professor, ISTD, SUTD

Agenda

- **Motivation**
- Course Logistics
- *“Grandfather Story”* on Database

Why Are You Here?

- You need to clear this if you are on…
  - Data Analytic track
  - Fin Tech track
  - Software Engineering track

Why Are You Here?

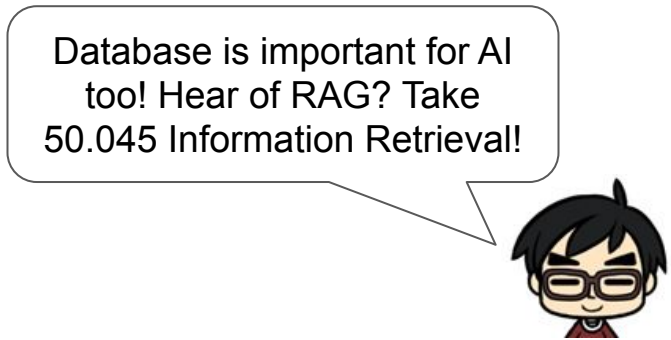
- You need to clear this if you are on…
  - Data Analytic track
  - Fin Tech track
  - Software Engineering track

Why Are You Here?

- Databases are part and parcel of software
- As software developers
  - Is the database design aligned the business logic?
  - Is it extendible and maintainable?
  - Why is this query running slow?

Why Are You Here?

- Databases are part and parcel of software
- As software developers
  - Is the database design aligned the business logic?
  - Is it extendible and maintainable?
  - Why is this query running slow?



What Do You Expect To Learn

- What a database system is
- How ~~hard~~ easy it is to design, use, and develop a database
- Big Data Systems: Spark & Hadoop
    - Cloud Computing: Amazon Web Service (AWS)

Measurable Outcomes

- Develop a database design for an application. (Week 1-3)
- List and explain major components of database and big data systems. (Week 1, 5-13)
- Write complex SQL queries. (Week 3-4)
- Estimate cost of different database operations. (Week 5-9)
- Compare different classes of big data systems. (Week 10-13)
- Write MapReduce and Spark jobs. (Week 10-13)
- Explain how a database differs to a big data system. (Week 5-13)
- Design, implement, and deploy database and big data systems on AWS. (Week 10-13)

What Can You Do At The End?

- Develop a database design for an application. (Week 1-3)
- List and explain major components of database and big data systems. (Week 1, 5-13)
- Write complex SQL queries. (Week 3-4)
- Estimate cost of different database operations. (Week 5-9)
- Compare different classes of big data systems. (Week 10-13)
- Write MapReduce and Spark jobs. (Week 10-13)
- Explain how a database differs to a big data system. (Week 5-13)
- Design, implement, and deploy database and big data systems on AWS. (Week 10-13)

What Can You Do At The End?

Data Owner: how do I store my data?

Database User: how do I use the data?

Database Designer: how do I build a database? (But you probably won't need to do it, instead understand why database performs this way.)

Agenda

- Motivation
- **Course Logistics**
- *"Grandfather Story"* on Database

Your Friendly Teaching Team

Instructors

Roy Ka-Wei Lee
(Week 1-4)
roy_lee@sutd.edu.sg

Zhang Wenxuan
(Week 5-9)
wxzhang@sutd.edu.sg

Qin Yanxia
(Week 10-13)
yanxia_qin@sutd.edu.sg

Teaching Assistants

- Zhengbo Zhang | zhengbo_zhang@mymail.sutd.edu.sg
- Tefera Addis Sisay | teferaaddis_sisay@mymail.sutd.edu.sg
- Do Viet Anh | doviet_anh@mymail.sutd.edu.sg

First thing First - Cohort 0!

- Check that you have the AWS Educate invitation email
- Please work on the AWS academy setup
- Check out Cohort class 0!

Course Structure

- 5 hours per week
  - 3 hour lecture (in LT)
    - Monday and Wednesday (1.5 hours each)
  - 2 hour tutorial/cohort sessions
- Many other hours learning on your own :x

Class Structure

- Slides will be made (available) **a moment before** the lectures
- Lectures might be recorded (best effort only - I try lah…)
- If you need clarification, or spot some bugs
  - ASK! - It will earn you participation mark!
- Tutorials/Cohorts: not graded
  - Spill-over content from lectures
  - Exercises, hands-on practice
  - Project consultation

Grading

- Homework/Assignment: 12% (2 sets, 6% each)
- Group project: 48%
  - Group of up to 3
  - Register your group member list by 08 June 2025, Sun 23:59 (refer to course handout)
- Participation: 3%
- Final: 37%, **Physical exam**

Academic Integrity

- **DO NOT CHEAT**:
  - Do not copy code/solutions from each other or online - WE WILL KNOW (quote from Prof Kenny Lu)!
- Do not put your solutions/projects online
- No late submission

Tips to do Well

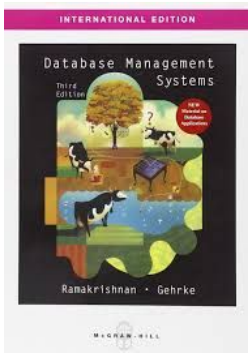
- DO NOT CHEAT!
- Complete your homework/assignment - It is designed to help you score points!
- Group project - be a team player and don’t devastated team mate (don’t be a *feeder* or *leaver*)
- Come for Cohorts! Participate in the exercises and discussion!
- Engage the friendly teaching team! We are here to help!

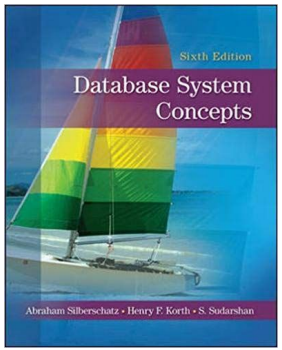
Schedule

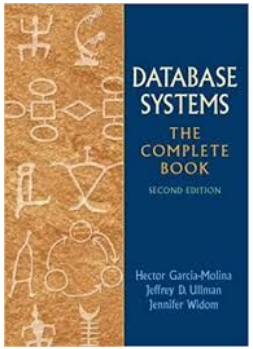
https://sutd50043.github.io/

Resources

- Main content comes from lectures
- Deeper details from books
- Exercises + hands-on from lab/tutorials
- No books for the “big-data” part.







stackoverflow

Google

OpenAI

Gemini

Acknowledgement

- Many pictures in the slides are taken from Internet (or AI generated)
- Some contents borrowed from:
  - MIT Database Systems (6.830)
  - University of Washington: Introduction to Data Management (CSE344)
  - CMU Database Systems (15-445/645)
  - ETH's Data Modeling and Databases (252-0063-00L)
  - ETH's Big Data For Engineers
  - Yale's Database System Concepts *Seventh Edition* (*https://codex.cs.yale.edu/avi/courses/CS-437/slides/index.html*)

Agenda

- Motivation
- Course Logistics
- ***"Grandfather Story"* on Database**

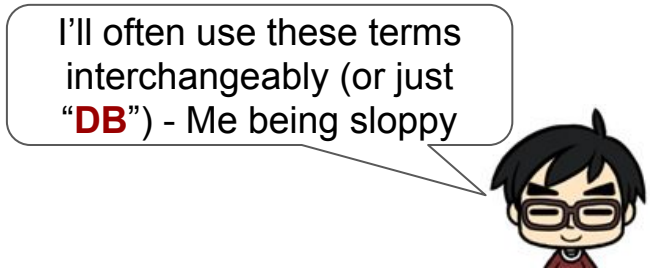
Learning Outcome

By the end of this lesson, you should be able to

- List the problems handled by database management systems
- Describe the techniques used in database system to solve these problems

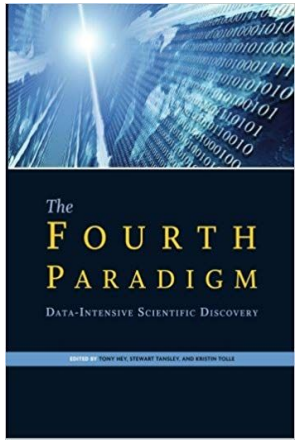
Database - What is it?

- What is a database?
  - Database is an **organized collection** of **data**
- What is a database management system (**DBMS**)?
  - System that **manages** the organized collection of data
    - Create, delete, store, query, analyze, etc.



Database - Why Should We Care?

- The world is drowning in data
- Changes the way we:
  - Make scientific discoveries
  - Live our lives (for better or for worse)
- Before that, you need to
  - Be able to manage data!



YouTube TikTok amazon NETFLIX

Database Applications Examples

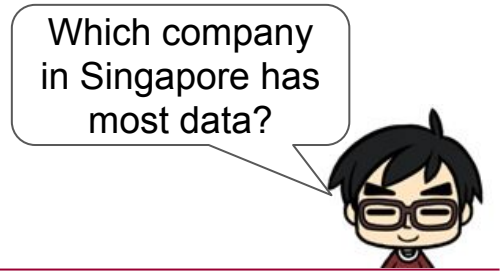
- **Enterprise Information**
  - Sales: customers, products, purchases
  - Accounting: payments, receipts, assets
  - Human Resources: Information about employees, salaries, payroll taxes.
- **Banking**
  - Customer information, accounts, loans, and banking transactions.
  - Credit card transactions
- **Finance**
  - Sales and purchases of financial instruments (e.g., stocks and bonds;
  - Storing real-time market data

Database Applications Examples

- **Universities**: registration, grades
- **Airlines**: reservations, schedules
- **Manufacturing**: management of production, inventory, orders, supply chain.
- **Telecommunication**: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- **Web-based services**
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- **Document databases**
- **Navigation systems**: for maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.

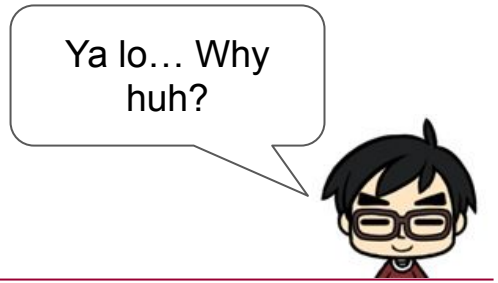
Database Applications Examples

- **Universities**: registration, grades
- **Airlines**: reservations, schedules
- **Manufacturing**: management of production, inventory, orders, supply chain.
- **Telecommunication**: records of calls, texts, and data usage, generating monthly bills, maintaining balances on prepaid calling cards
- **Web-based services**
  - Online retailers: order tracking, customized recommendations
  - Online advertisements
- **Document databases**
- **Navigation systems**: for maintaining the locations of various places of interest along with the exact routes of roads, train systems, buses, etc.



Why Do We Need a Specialized DB

- Why not build the DB directly on top of file systems.
- Use the OS file system
- Write the programs accessing the data using a regular PL
  - C,
  - Java,
  - Python



Early Database Systems

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues

Early Database Systems

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues
  - **Data redundancy and inconsistency**
    - Data is stored in multiple file formats resulting in duplication of information in different files

Early Database Systems

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues
  - **Data redundancy and inconsistency**
    - Data is stored in multiple file formats resulting in duplication of information in different files
  - **Difficulty in accessing data**
    - Need to write a new program to carry out each new task

Early Database Systems

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues
  - **Data redundancy and inconsistency**
    - Data is stored in multiple file formats resulting in duplication of information in different files
  - **Difficulty in accessing data**
    - Need to write a new program to carry out each new task
  - **Integrity problems**
    - Integrity constraints (e.g., account balance > 0) become “buried” in program code rather than being stated explicitly
    - Hard to add new constraints or change existing ones

Early Database Systems (Cont.)

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues
  - **Atomicity of updates**
    - Failures may leave database in an inconsistent state with partial updates carried out
    - Example: Transfer of funds from one account to another should either complete or not happen at all

Early Database Systems (Cont.)

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues
  - **Atomicity of updates**
    - Failures may leave database in an inconsistent state with partial updates carried out
    - Example: Transfer of funds from one account to another should either complete or not happen at all
  - **Concurrent access by multiple users**
    - Concurrent access needed for performance
    - Uncontrolled concurrent accesses can lead to inconsistencies
    - Ex: Two people reading a balance (say 100) and updating it by withdrawing money (say 50 each) at the same time

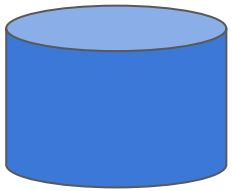
Early Database Systems (Cont.)

- Long before your time… database applications were built directly on top of file systems, which leads to all sorts of issues
  - **Protection problems**
    - Hard to provide user access to some, but not all, data

Early Database Systems (Summary)

File System DB Issues

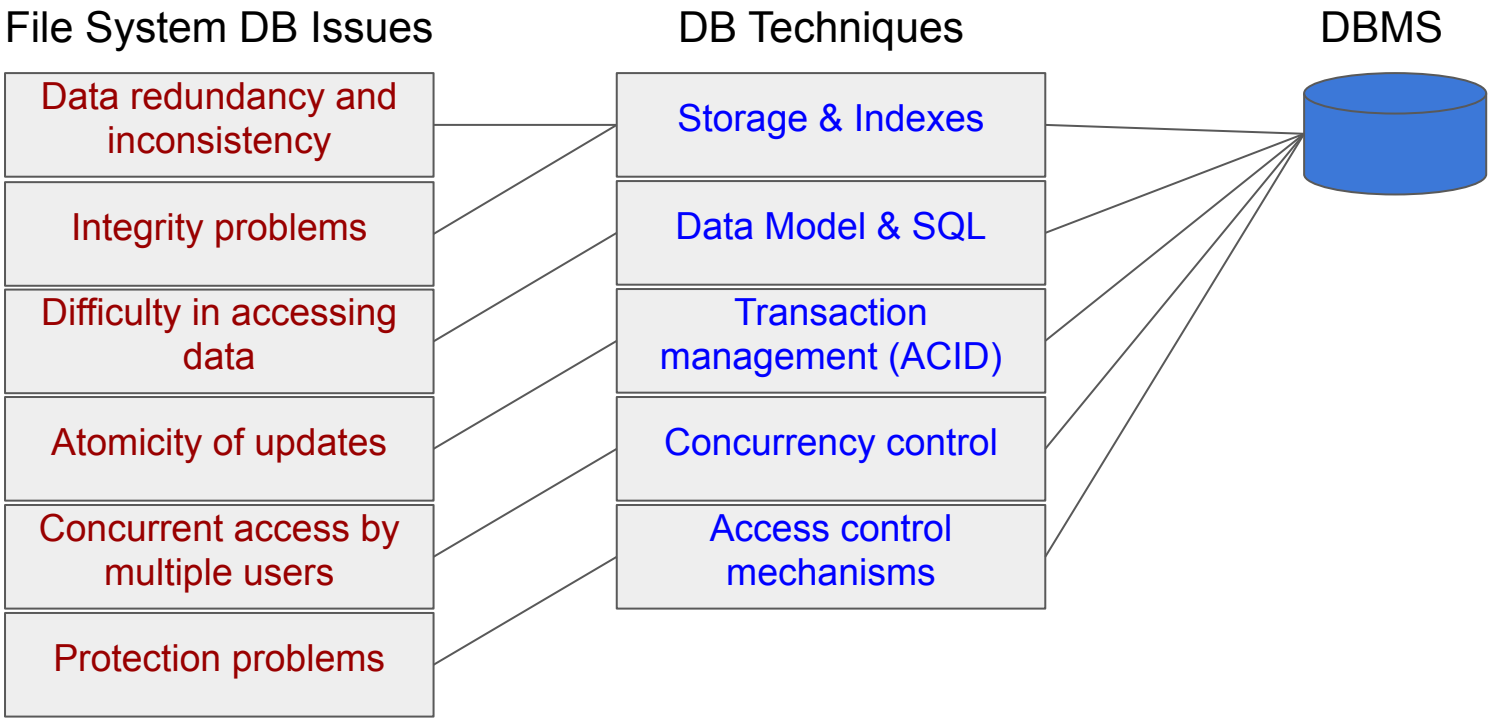
Data redundancy and inconsistency	Difficulty in accessing data	Integrity problems
Atomicity of updates	Concurrent access by multiple users	Protection problems



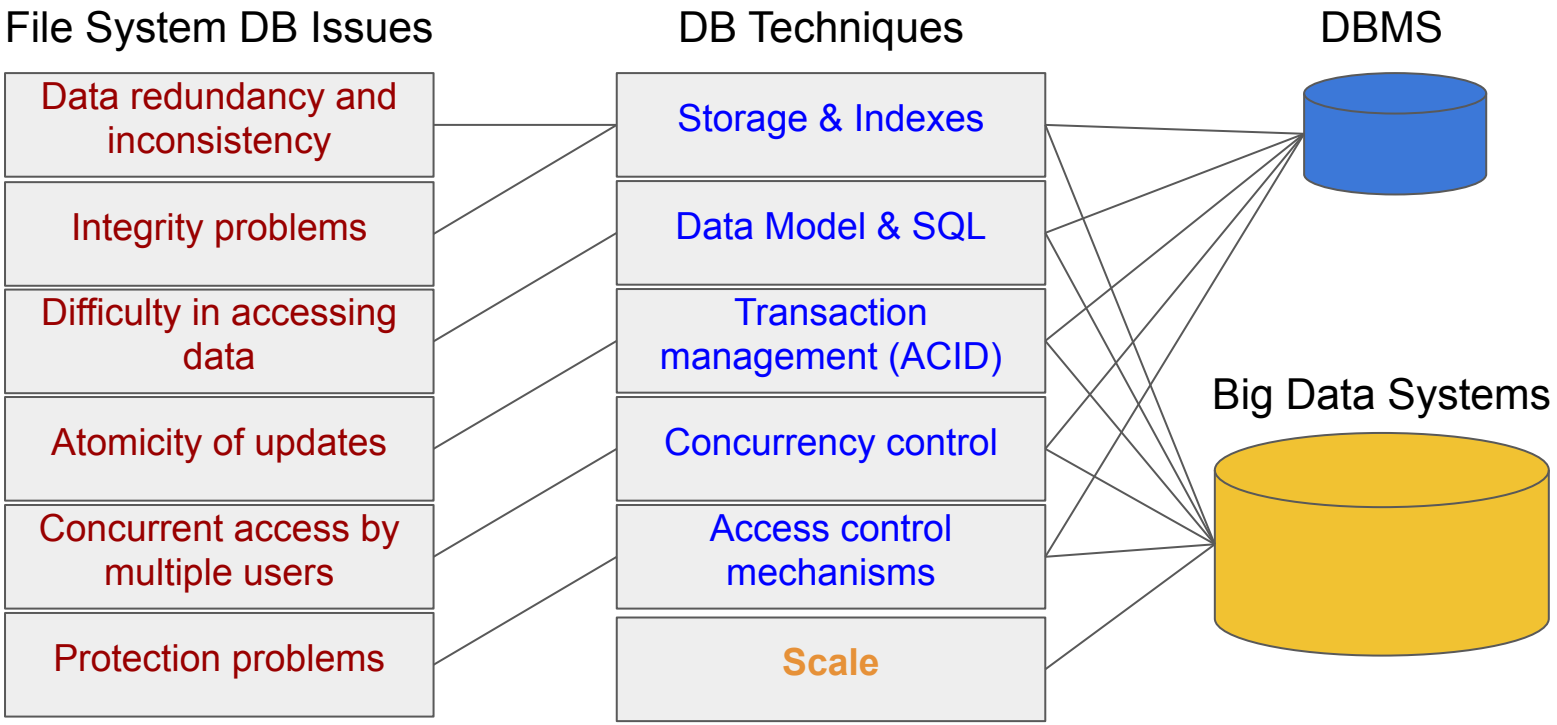
DBMS

Modern Database systems offer solutions to all the above problems

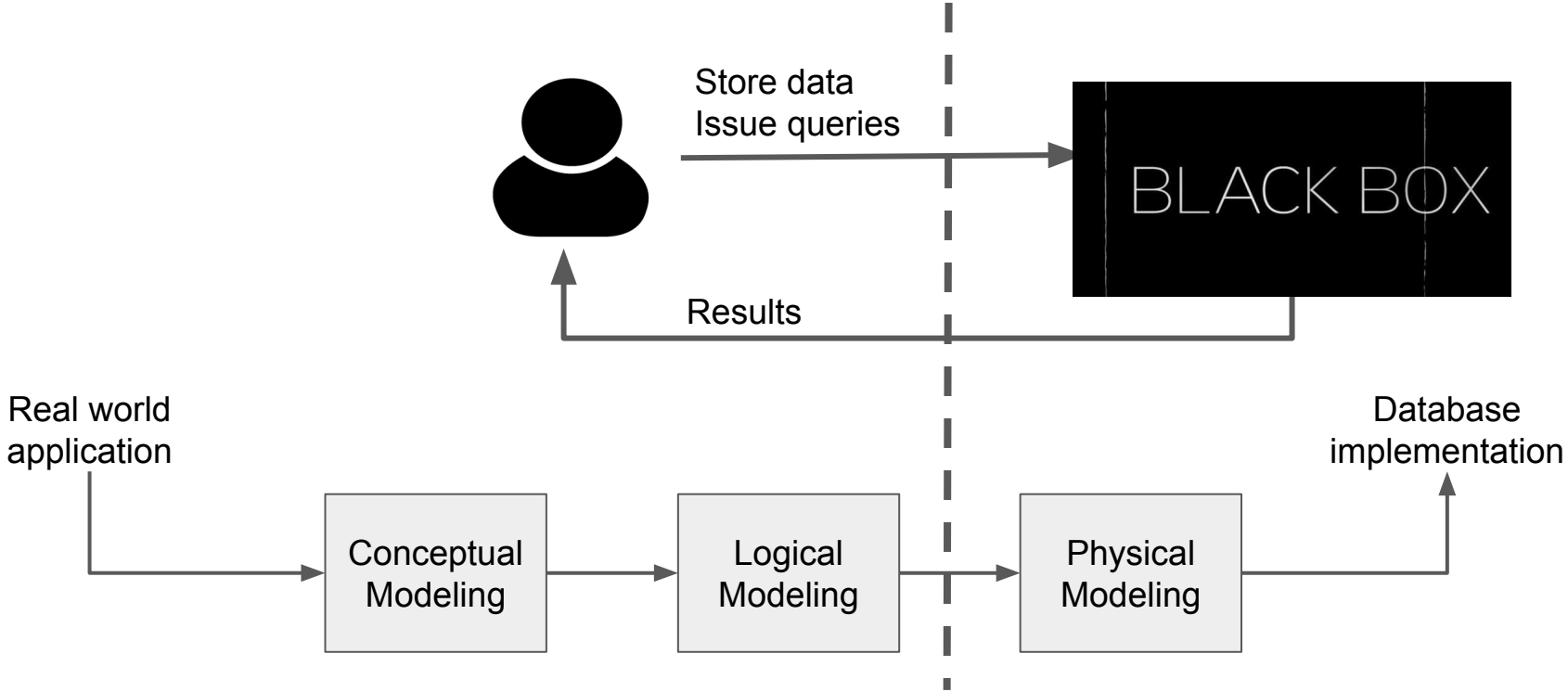
How DBMS Solve These Problems



How DBMS Solve These Problems



Overview of Data Modeling



History of Database Systems

- 1950s and early 1960s:
  - Data processing using magnetic tapes for storage
    - Tapes provided only sequential access
  - Punched cards for input

History of Database Systems

- Late 1960s and 1970s:
  - Hard disks allowed direct access to data
  - Network and hierarchical data models in widespread use
  - Ted Codd defines the relational data model
    - Won the ACM Turing Award for this work
    - IBM Research begins System R prototype
    - UC Berkeley (Michael Stonebreaker) begins Ingres prototype
    - Oracle releases first commercial relational database
  - High-performance (for the era) transaction processing

History of Database Systems

- 1980s:
  - Research relational prototypes evolve into commercial systems
    - SQL becomes industrial standard
  - Parallel and distributed database systems
    - Wisconsin, IBM, Teradata
  - Object-oriented database systems
- 1990s:
  - Large decision support and data-mining applications
  - Large multi-terabyte data warehouses
  - Emergence of Web commerce

History of Database Systems

- 2000s
  - Big data storage systems
    - Google BigTable, Yahoo PNuts, Amazon,
    - “NoSQL” systems.
  - Big data analysis: beyond SQL
    - Map reduce and friends
- 2010s
  - SQL reloaded
    - SQL front end to Map Reduce systems
    - Massively parallel database systems
    - Multi-core main-memory databases

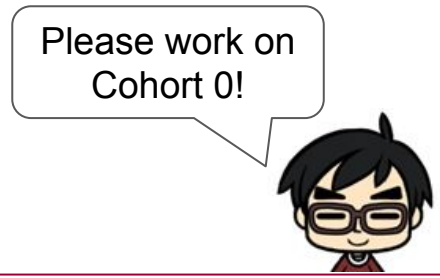
History of Database Systems

- 2020s
  - AI-native and vector databases
    - Rise of purpose-built databases for machine learning, embeddings, and retrieval-augmented generation (e.g., **Pinecone**, **Weaviate**, **FAISS**)
  - Cloud-native, serverless, and autoscaling databases
    - Databases as services (e.g., **Google Spanner**, **AWS Aurora Serverless**, **CockroachDB**) with elastic scaling, high availability, and global distribution
  - Real-time, streaming-first architectures
    - Unified batch and stream processing (e.g., **Apache Flink**, **Materialize**, **Delta Lake**

What you should know?

- What are the problems handled by database management systems?
- How database system to solve these problems?

Reading Resources:
- https://sutd50043.github.io/notes/l1_intro/



Acknowledgement

- *The following material have been referenced or partially used:*
  - *MIT Database Systems (6.830)*
  - *University of Washington: Introduction to Data Management (CSE344)*
  - *CMU Database Systems (15-445/645)*
  - *ETH's Data Modeling and Databases (252-0063-00L)*
  - *ETH's Big Data For Engineers*
  - *Yale's Database System Concepts Seventh Edition (https://codex.cs.yale.edu/avi/courses/CS-437/slides/index.html)*