THE UNIVERSITY of EDINBURGH

Advanced Database Systems

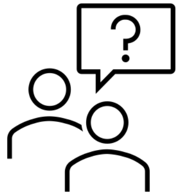
Spring 2025

Lecture #01:

Course Introduction

1

Essential Questions



Why take this course?

What is this course about?

Who is running this course?

How will this course work?

2

Why? Reason #1: Utility

Data processing backs essentially every application

Databases of one form or another back most applications

The **principles** taught in this course back nearly everything in computing

Knowing how to manage data is a vital, core asset in today's world

This material will empower you as a computer scientist

3

Why? Reason #2: Centrality

Data is at the **centre** of modern society

Much cheaper to generate data

- Sensors, smart devices, social networks, online games, software logs, audio & video

Much cheaper to process data

- Cloud computing, open-source software, heterogenous architectures (CPU, GPU, FPGA)

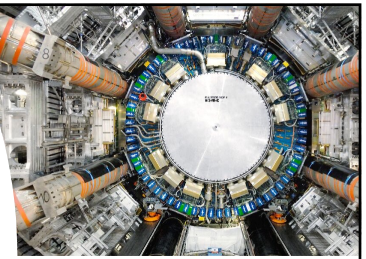
4

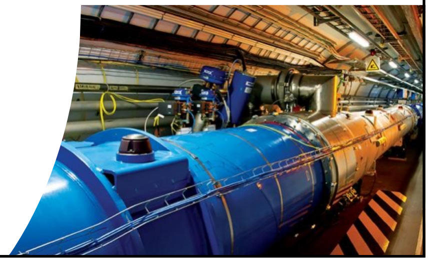
How much data is generated *every minute?*

https://www.domo.com/learn/data-never-sleeps-11
(2023 edition)

5

Scale of Scientific Data





Large Hadron Collider, CERN

Raw data: **600,000,000 GB/sec**
(19 ZettaBytes/year) Zetta = 10^{21}

Downsampled: **25GB/sec**
(788 PetaBytes/year) Peta = 10^{15}

Downsampled further: **1050MB/sec**
(33 PetaBytes/year)

https://home.cern/science/computing/processing-what-record

6

Why? Reason #2: Centrality

Data is at the **centre** of modern society

Much cheaper to generate data
- Sensors, smart devices, social networks, online games, software logs, audio & video

Much cheaper to process data
- Cloud computing, open-source software, heterogenous architectures (CPU, GPU, FPGA)

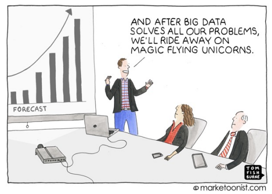
The infrastructure determines what's possible

7

Why? Reason #3: The Core of Computing

Data growth will continue to outpace computation
- Philosophy: more data → more value?

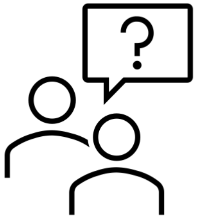
Systems for managing data at scale: the core of modern computing



Techniques you learn in this course underlie many topics in computing

8

Essential Questions



Why take this course?

What is this course about?

Who is running this course?

How will this course work?

What is a Database?

A database is an organised collection of inter-related data that models some aspect of the real world

Databases are the core component of most computer applications

- Banking
- Web and mobile apps
- Online retailers
- Human resources

Sometimes confused with a Database Management System

What is a DBMS?

A database management system (DBMS) is software that **stores**, **manages**, and **facilitates** access to databases

- Mediates interactions between users and databases

Traditionally, DBMS refers to relational databases

- SQL, ACID transactions, prevent data loss
- This will be the focus of this course!

Warning: market and terms in rapid transition

- The tech remains (roughly) the same
- Good time to focus on fundamentals!

ORACLE

Microsoft SQL Server

MySQL

SQLite

PostgreSQL

Why using a DBMS?

Consider one typical scenario:

1. Create a database that models a university organisation to keep track of students, instructors, and courses
2. Build an application to support typical operations on the DB:
   - Add new students, instructors, and courses
   - Register students for courses and generate class rosters
   - Assign grades to students, compute GPA, and generate transcripts

Flat File Strawman

Store our database as comma-separated value (CSV) files

Instructor(name, dept, salary)

```
"Jones", "CS", 95000
"Smith", "Physics", 75000
"Gold", "CS", 62000
```

instructor.csv

Course(name, instructor, year)

```
"Databases", "Jones", 2018
"Quantum M.", "Smith", 2017
"Compilers", "Jones", 2017
```

course.csv

Apps have to parse the files each time they want to read/update records

13

Flat File Strawman

Example: Get the names of all computer science instructors

Instructor(name, dept, salary)

```
"Jones", "CS", 95000
"Smith", "Physics", 75000
"Gold", "CS", 62000
```

instructor.csv

```
for line in file:
  record = parse(line)
  if "CS" == record[1]:
    print record[0]
```

Tight coupling between application logic and physical storage

14

Flat File: Drawbacks

Data redundancy

- Duplication of information in different files
- Ex: changing string "CS" to "Computer Science" requires rewriting several files

Storage format needs to be exposed

- Developers need to be aware of the physical layout of data
- Data may be stored in various file formats such as CSV, JSON, binary, etc.

Difficulty in accessing data

- Need to write a new program to carry out each new task
- Programming complex logic on several files can be error-prone and inefficient

15

Flat File: Drawbacks (Cont.)

Search is expensive (no indexes)

- Cannot find tuple with given key quickly
- Always have to read the entire file

No atomicity of updates

- Failures may leave database in an inconsistent state with partial updates carried out
- Ex: moving money between two accounts should either complete or not happen at all

Integrity problems

- Integrity constraints (e.g., course mark must be ≥ 0) become "buried" in program code
- Hard to add new constraints or change existing ones

16

Flat File: Drawbacks (Cont.)

Concurrent access by multiple users

- Concurrent access needed for performance
- Uncontrolled concurrent accesses can lead to inconsistencies

No security

- Hard to provide user access to some, but not all, data
- Storing data in raw CSV files is insecure

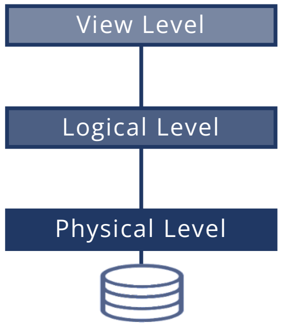
No application programming interface

- How can a payroll program access the data?

Database systems offer solutions to all the above problems

17

Levels of Abstractions



View Level: Simplifies interaction with the database, hides info (e.g., salary) for security purposes

Logical Level: Describes data stored in the DB

```
type instructor = record
  name: string;
  dept: string;
  salary: integer;
end
```

Physical Level: Describes how a record is stored

Data independence:
Insulate users from changes in lower levels

18

Data Models

Data model

- Collection of concepts for describing the data in a database

Schema

- Description of a particular collection of data, using a given model

Models in practice

- Relational, key-value, graph, document, array, hierarchical, network

Most DBMSs implement the relational data model

19

Ranking of DBMS Technologies 2025

423 systems in ranking, January 2025

Rank Jan 2025	Rank Dec 2024	Rank Jan 2024	DBMS	Database Model	Score Jan 2025	Score Dec 2024	Score Jan 2024
1.	1.	1.	Oracle	Relational, Multi-model	1258.76	-5.03	+11.27
2.	2.	2.	MySQL	Relational, Multi-model	998.15	-5.61	-125.31
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	798.55	-7.14	-78.05
4.	4.	4.	PostgreSQL	Relational, Multi-model	663.41	-2.97	+14.45
5.	5.	5.	MongoDB	Document, Multi-model	402.50	+2.12	-14.98
6.	↑ 7.	↑ 9.	Snowflake	Relational	153.90	+6.54	+27.98
7.	↓ 6.	↓ 6.	Redis	Key-value, Multi-model	153.36	+3.08	-6.03
8.	8.	↓ 7.	Elasticsearch	Multi-model	134.92	+2.60	-1.15
9.	9.	↓ 8.	IBM Db2	Relational, Multi-model	122.97	+0.19	-9.43
10.	10.	↑ 11.	SQLite	Relational	106.69	+4.97	-8.51
11.	11.	↑ 12.	Apache Cassandra	Wide column, Multi-model	99.19	+1.26	-11.84
12.	12.	↓ 10.	Microsoft Access	Relational	92.70	+1.88	-24.97
13.	13.	↑ 17.	Databricks	Multi-model	87.85	+0.16	+7.31

Based on #mentions (e.g., stack overflow), google trends, job postings, profile data on LinkedIn, tweets…

http://db-engines.com/en/ranking

21

What Is This Course About?

Big ideas in database management systems

- **Principles**: data independence, declarative programming, isolation, consistency
- **Core algorithms**: search, optimisation, evaluation, concurrency
- **System designs**: how to compose components into a technological stack

The heart of scalable computer systems

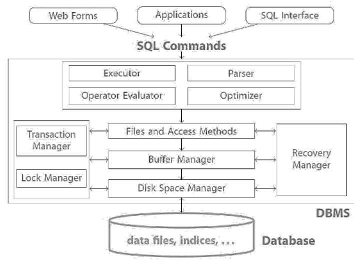
Many of the details and technologies will change in the future

- Be prepared to generalize from what you learn here
- Keep learning new things

What Is This Course About?

Design and **implementation** of disk-oriented DBMSs

- Storage and file structure
- Indexing techniques
- Query evaluation (theory & practice)
- Query optimisation
- Transaction management
- Distributed and parallel databases



Learning Outcomes

Gain insights into how DBMSs function internally

Learn data management techniques that can help **YOU, the future scientist,** to transform data into knowledge and build new DBMS technologies

Distinguish “hard” vs. “easy” in query evaluation

Learn fundamental concepts used in CS and beyond

Course Prerequisites

Recommended: Introductory course on Databases

- Developing applications using relational DBMSs
- Good knowledge of query languages is a plus ⇐ **We will briefly revisit them**

Design and analysis of algorithms

- Sorting & searching algorithms, big-O notation

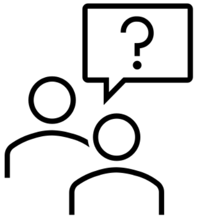
Basic familiarity with complexity

- PTIME, NP-complete

Solid programming skills

- Coursework includes one programming assignment in Java

Essential Questions



Why take this course?

What is this course about?

Who is running this course?

How will this course work?

27

Who Is Running This Course?

[Milos Nikolic](#)

Lecturer, School of Informatics

Interests: database systems, in-database machine learning, stream processing, query compilation

[Andreas Pieris](#)

Reader, School of Informatics

Interests: database theory, knowledge-enriched data, knowledge representation and reasoning

28

How Will This Course Work?

In-person lectures — weeks 1-10

- All lectures are live streamed and recorded for later viewing
- Check the course schedule and timetable for more information
- Lectures are followed by **short online quizzes**

In-person tutorials — weeks 3, 5, 7, 9, 11

- Discussing your answers to tutorial sheets
- To change your tutorial group, use the group change request form

No practical labs

29

Lecture Overview

Block 0: Databases and Query Languages — week 1, Milos

- Crash course on SQL and relational algebra
- Covered in an introductory database course

Block 1: DBMS Internals — weeks 2-8, Milos

- How to implement different parts of a database system?
- Important for the coursework assignment

Block 2: Theory of Query Evaluation — weeks 9-10, Andreas

- This is not a theory database course…
  - … but understanding the fundamentals is essential for implementation

30

Assessment Structure

Programming assignment (40%)

- Implement features in an educational database system in Java

Course engagement (10%)

- Weekly online quizzes

Final exam (50%)

- In-person exam

School of Informatics uses a Common Marking Scheme

- 1st class or MSc distinction: 70% and above

Programming Assignment

Involves coding in Java

Requires good programming skills

- Java expertise is not mandatory
- But experience with object-orient programming is expected

Released in week 2

- Some topics covered by then, others covered later
- Allows you to start early & better manage your time

Due: Thursday, 27 March @ 12 noon

Online Quizzes

Short online quizzes released after each lecture

Goals: engage & reinforce the basics

Marking rules

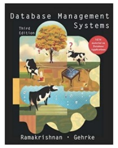
- Quizzes are auto-marked on Learn
- **2 attempts** for each quiz (higher mark counts)
- Each quiz counts equally for engagement
- No deadline per quiz. **Latest submission is Thursday, 3 April @ 12 noon**

Max engagement mark is 100

Textbooks

Database Management Systems, 3rd edition
Ramakrishnan and Gehrke

- Most lectures will closely follow this book
- Old edition (2003) but still relevant and unbeatable



Principles of Databases, preprint
Barcelo, Arenas, Libkin, Martins, and Pieris

- Comprehensive material on database theory
- https://github.com/pdm-book/community

Recent Course Changes

- Different exam format since 2022/23
  - 6-8 smaller questions, all mandatory
  - Same format this year
- Coursework assignment released early
  - Self-assess if your programming skills suffice before the end of week 2
- Course content similar as in previous years
  - This 20-credit course replaces Advanced Databases (INFR11011)
  - Content from INFR11011 (e.g., exam questions) still relevant
  - **NEW in 2024/25:** New topics (WCOJ, NoSQL), some not covered in depth (recovery)

Response to Last Year's Course Feedback

- Some topics relevant for CW were covered late in the course
  - We have adjusted the schedule to cover relevant topics early on
- Can we cover other types of database systems (e.g., No SQL)?
  - We have reserved one lecture to cover the basic principles of NoSQL systems
- Provide skeleton code for CW
  - We will provide you with boilerplate code to allow you to focus on important aspects

Plagiarism Policy

- All assignments must be your own work
- They are **not** group assignments
- You may **not** copy source code from other people or the web
- You may **not** use public repositories to host your code
- We have the technology to detect cheating
- See UoE Academic misconduct for more information

Staying in Touch

- All class communication via Piazza
- Announcements and discussion
  - Read it regularly
  - Post all questions/comments there
  - Answer each other's questions!
- Piazza's Live Q&A for asking questions while watching the live stream
- **Sign up now** on Learn

Acknowledgement

The lecture slides in this course incorporate content from various individuals, to which I am grateful:

- D. Olteanu (Zurich)
- T. Furche (Oxford)
- J. Hellerstein (Berkeley)
- A. Pavlo (CMU)
- T. Grust (Tübingen)
- R. Ramakrishnan (Microsoft)
- J. Gehrke (Microsoft)