THE UNIVERSITY of EDINBURGH

Advanced Database Systems

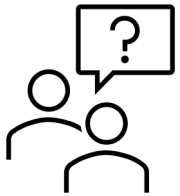
Spring 2026

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

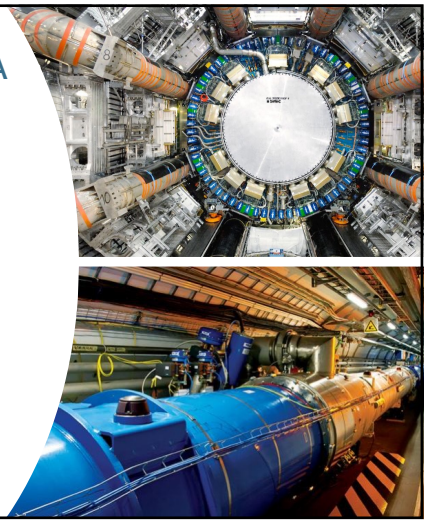
Large Hadron Collider, CERN

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

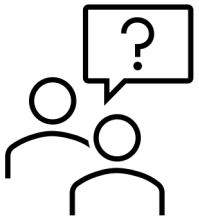
Philosophy: more data → more value?

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?

9

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

10

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

11

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

12

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

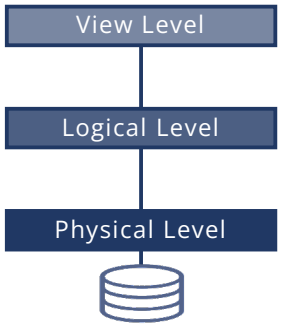
- Hard to provide user access to some, but not all, data
- Storing data in raw CSV files is insecure

No application programming interface

- How can a payroll program access the data?

Database systems offer solutions to all the above problems

Levels of Abstractions



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

Describes data stored in the DB

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

Describes how a record is stored

Data independence:
Insulate users from changes in lower levels

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

Ranking of DBMS Technologies 2026

428 systems in ranking, January 2026

Rank Jan 2026	Rank Dec 2025	Rank Jan 2025	DBMS	Database Model	Score Jan 2026	Score Dec 2025	Score Jan 2025
1.	1.	1.	Oracle	Relational, Multi-model	1237.34	+2.94	-21.42
2.	2.	2.	MySQL	Relational, Multi-model	867.52	-0.97	-130.63
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model	706.26	-16.26	-92.30
4.	4.	4.	PostgreSQL	Relational, Multi-model	666.27	+6.84	+2.86
5.	5.	5.	MongoDB	Multi-model	376.74	+4.46	-25.77
6.	6.	6.	Snowflake	Relational	207.79	+5.34	+53.89
7.	7.	7.	Redis	Key-value, Multi-model	144.16	+1.68	-9.20
8.	8.	↑ 13.	Databricks	Multi-model	141.55	+3.53	+53.70
9.	9.	9.	IBM Db2	Relational, Multi-model	112.72	-3.04	-10.25
10.	10.	↓ 8.	Elasticsearch	Multi-model	107.15	-2.68	-27.78
11.	↑ 12.	11.	Apache Cassandra	Wide column, Multi-model	100.84	-0.71	+1.65
12.	↓ 11.	↓ 10.	SQLite	Relational	100.61	-2.01	-6.08
13.	13.	↑ 14.	MariaDB	Relational, Multi-model	87.72	+0.13	+2.14

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

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

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

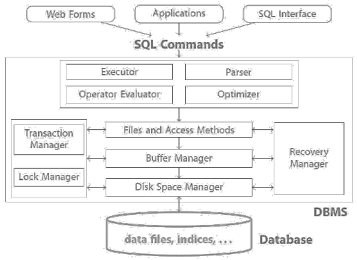
Many of the details and technologies will change in the future

- 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

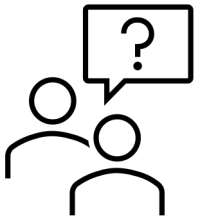
Basic familiarity with complexity

- 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

Reader, School of Informatics

Interests: database systems, in-database machine learning, stream processing, graph processing

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**

NEW: In-person revision classes — weeks 3, 5, 7, 9

- Whole class, tutorial-style discussions
- Replaces traditional tutorials to encourage greater engagement

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 (50%)

Implement features in an educational database system in Java

Final exam (50%)

In-person exam

Formative activities

Weekly online quizzes + revision classes

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, 19 March @ 12 noon

Textbooks

Database Management Systems, 3rd edition
Ramakrishnan and Gehrke

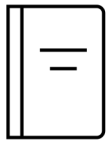
Most lectures will closely follow this book

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

NEW: One A4 page of notes permitted (double-sided)

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-class revision classes in weeks 3, 5, 7, 9

Student Feedback from Previous Years

- 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
- Advice for current students
  - **Start early with CW**

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)