

IN3020&4020 – Database Systems (2022)

Part: Intro & Recap

Lecture 1: Introduction and Motivation

17 January

Egor V. Kostylev

IFI, University of Oslo

egork@ifi.uio.no



About myself

2004–2009: PhD, Lomonosov Moscow State University
Programming Languages

2010–2013: PostDoc, University of Edinburgh
Databases

2013–2020: Lecturer, University of Oxford
+ Knowledge Representation

from **Sep. 2020:** Associate Professor, University of Oslo
+ Hybrid AI

Language: Jeg snakker litt **norsk**, but the course is in **English**

How to find me: egork@ifi.uio.no, office 8165 in OJD

Digital (in *Zoom*) for some time (**2 weeks?**):

- challenging for both **you** and **me**

I need your **cooperation**:

- please, turn **video** on if possible
- ask **questions** (rise 'yellow' hand)
- I will also ask questions time to time

Lectures will be **recorded**:

- videos will be **published** (very) soon after the lecture
- do not forget about **mandatories** (see below)

What this is about hierarchy

IN2090 – Databaser og datamodellering (Leif Harald Karlsen)

- What databases are and how to use them
- pre-requisite (obligatoriske forkunnskaper) for IN3020
- not for IN4020 (knowledge of the material is still expected)

IN3020&4020 – Database systems

- How databases work
- pre-requisite for IN5040

IN5040 – Advanced database systems (Vera Hermine Goebel)

- New challenges in data management

Our *general strategy* for IN2090 material:

- Not important for us: ignored
- We build upon: presented, but quickly
- We study deeper: **repeated in detail** (and extended)

Our use of IN2090 synopsis:

1. *Data modelling*

- **relational model** (including keys)
- **relational algebra**
- entity-relation (ER) diagrams
- normal forms and decomposition

2. *SQL*

- basic and **complex** operations
- **security in DBs**
- **indexing and query processing**

Databases and Database Management Systems

Database:

- collection of related **data**—that is, known facts with implicit meaning—organised in some way (**data model**)
- *In this course:* **relational databases** (mostly), **RDF knowledge graphs**, etc.

Database management system (DBMS):

- a **software** that facilitates **defining**, **constructing**, **manipulating**, and **sharing** databases by various users and applications
- *In this course:* **SQL engines** (mostly), **SPARQL engines**, etc.

We often use ‘database’ and ‘DBMS’ **interchangeably**, as well as **omit** ‘relational’ (when this does not cause confusion, which is usually the case)

Why to bother about databases?

CSV + Python:

```
import csv
import os
filea="a.csv"
fileb="b.csv"
temp="temp.csv"
src1=csv.reader(open(filea,"r"),delimiter=",")
src2=csv.reader(open(fileb,"r"),delimiter=",")
src2_dict={}
for row in src2:
    src2_dict[row[0]] = row[1]
with open(temp,"w") as fout:
    csvwriter=csv.writer(fout, delimiter=delim)
    for row in src1:
        if row[1] in src2_dict:
            row[3]=src2_dict[row[1]]
        csvwriter.writerow(row)
os.rename(temp, filea)
```

Database + SQL:

```
UPDATE a
SET c4=b.c2
FROM b
WHERE a.c2=b.c1;
```

Question: Why the right is better than the left?

Why to bother about (dedicated) databases?

CSV + Python:

```
...  
for row in src2:  
    src2_dict[row[0]] = row[1]  
...
```

Database + SQL:

```
UPDATE a  
SET c4=b.c2  
...
```

Question: Why the right is better than the left?

Answer 1: SQL is easier to learn than Python

Answer 2: SQL is **declarative**:

- we tell the engine **what we want** rather than **how to compute it**
- **easier** (and shorter) to formulate
- much more space for **optimisation**:
 - rewrite query into a more efficient one
 - store data in an easy-to-access way
 - efficient treatment of many users with complex transactions

Why we concentrate on relational databases?

Relational databases (with SQL):

- are the **most popular** database architecture
- have good **balance** between simplicity and efficiency
- **natural** and **intuitive** to work with

‘Databases’ usually mean ‘relational databases’

For us, relational databases is just a **typical example**, and many challenges and approaches we will study are applicable to other database architectures

In the end of the course, we will look at other, **NoSQL** architectures:

- RDF + SPARQL, Neo4j + Cypher, MapReduce + Hadoop

Inside the database 'black box'

- What is apparent to us is that we **write** an SQL query, and the database **does something** and comes up with an answer
- But there is a lot that happens **behind the scenes**, inside the **'black box'**
- This course will show what happens behind the scenes in the **'black box'**, so that we understand the common **challenges** and what to do about them

Challenge 1: Query optimisation (example)

```
SELECT B, C, Y
FROM R, S
WHERE W = X AND A = 3 AND Z = 'a'
```

Naive approach:

cross each tuple in **R** with each in **S**; select using condition; project

Relation R

A	B	C	...	W
1	z	1	...	4
2	c	6	...	2
3	r	8	...	7
4	n	9	...	4
2	j	0	...	3
3	t	5	...	9
7	e	3	...	3
8	f	5	...	8
1	h	7	...	5

Relation S

X	Y	Z
1	a	a
2	f	c
3	t	b
4	b	b
7	k	a
6	e	a
7	g	c
8	i	b
9	e	c

~>

A	B	C	...	W	X	Y	Z
1	z	1	...	4	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	...
2	c	6	...	2	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	...
3	r	8	...	7	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	...
3	r	8	...	7	7	k	a
4	n	9	...	4	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	...
...	...	...	...	...	...	...	v
2	i	0	...	3	1	a	a
...	...	...	...	...	2	c	c
...	...	...	...	...	...	...	...
3	t	5	...	9	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	...
7	e	3	...	3	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	v

~>

B	C	Y
r	8	k

Challenge 1: Query optimisation (example)

```
SELECT B, C, Y
FROM R, S
WHERE W = X AND A = 3 AND Z = 'a'
```

Question: Can we do it smarter?

Relation R

A	B	C	...	W
1	z	1	...	4
2	c	6	...	2
3	r	8	...	7
4	n	9	...	4
2	j	0	...	3
3	t	5	...	9
7	e	3	...	3
8	f	5	...	8
1	h	7	...	5

Relation S

X	Y	Z
1	a	a
2	f	c
3	t	b
4	b	b
7	k	a
6	e	a
7	g	c
8	i	b
9	e	c

→

A	B	C	...	W	X	Y	Z
1	z	1	...	4	1	a	a
...	...	...	...	...	2	f	c
2	c	6	...	2	1	a	a
...	...	...	...	...	2	f	c
3	r	8	...	7	1	a	a
...	...	...	...	...	2	f	c
3	r	8	...	7	7	k	a
4	n	9	...	4	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	v
2	i	0	...	3	1	a	a
...	...	...	...	...	2	c	c
...	...	...	...	...	...	...	...
3	t	5	...	9	1	a	a
...	...	...	...	...	2	f	c
7	e	3	...	3	1	a	a
...	...	...	...	...	2	f	c
...	...	...	...	...	...	...	v

→

B	C	Y
r	8	k

Challenge 1: Query optimisation (example)

```
SELECT B, C, Y
FROM R, S
WHERE W = X AND A = 3 AND Z = 'a'
```

Smarter approach:

select relevant tuples in **R** and **S**; cross with selecting; project

Relation R

A	B	C	...	W
1	z	1	...	4
2	c	6	...	2
3	r	8	...	7
4	n	9	...	4
2	j	0	...	3
3	t	5	...	9
7	e	3	...	3
8	f	5	...	8
1	h	7	...	5

Relation S

X	Y	Z
1	a	a
2	f	c
3	t	b
4	b	b
7	k	a
6	e	a
7	g	c
8	i	b
9	e	c



Relation R

A	B	C	...	W
1	z	1	...	4
2	c	6	...	2
3	r	8	...	7
4	n	9	...	4
2	j	0	...	3
3	t	5	...	9
7	e	3	...	3
8	f	5	...	8
1	h	7	...	5

Relation S

X	Y	Z
1	a	a
2	f	c
3	t	b
4	b	b
7	k	a
6	e	a
7	g	c
8	i	b
9	e	c



A	B	C	...	W	X	Y	Z
3	r	8	...	7	7	k	a



B	C	Y
r	8	k

Challenge 2: Efficient data storage (example)

```
SELECT *  
FROM R  
WHERE A = 3
```

Relation R

A	B	C	...	W
1	z	1	...	4
2	c	6	...	2
3	r	8	...	7
4	n	9	...	4
2	j	0	...	3
3	t	5	...	9
7	e	3	...	3
8	f	5	...	8
1	h	7	...	5

Naive approach: Try each tuple one by one, select relevant

Question: Can we represent data smarter so trying all is usually not necessary?

A solution: Sort tuples in the table in advance according to **A**, use binary search when evaluating the query

Problem: Works only for **A**, what if we also expect queries for **W**?

Better solution: Use *indexes*! (See the course)

Challenge 3: Efficient transaction management

Instructions over databases come in sequences, called **transactions**

- 'instruction' may be a **query** (update, select, etc.) or a finer **piece of a query** (e.g., read the tuple by this pointer, add this tuple at the end of this table, etc.)
- each transaction is expected to be evaluated **in full** ('all or nothing') to preserve consistency
- transactions may come in parallel (e.g., from different users)

Requirements can be formalised as **ACID** principles:

Atomicity, Consistency, Isolation, Durability

Approach:

Transaction Management (locking, logging, buffering, etc.)

At the end of the course you will know

- How DBMS work inside (relational and NoSQL)
- How queries are optimised and evaluated
- How data is stored and managed
- ACID principles
- Transaction management types (i.e., isolation levels)
- Theoretical and practical aspects of database administration

Three main parts:

- Intro and SQL recap
- 1. Query processing in relational databases
 - relational algebra, indexes, query optimisation and evaluation
- 2. Transaction management in relational databases
 - ACID, serialisation, concurrency control, isolation levels
- Advanced DBMS Architecture (for IN4020)
- 3. NoSQL DBMS
 - knowledge graphs, column-based, etc.
- A bit about emerging technologies and research

Lecture schedule – 1 (preliminary)

#	Date	Part	Topic
1	17.01 (Mon)	Intro+SQL	Introduction
2	20.01 (Thu)	Intro+SQL	Relational Model
3	24.01 (Mon)	Intro+SQL	SQL recap – 1
4	27.01 (Thu)	Intro+SQL	SQL recap – 2
5	31.01 (Mon)	Part 1: Query Processing	Relational Algebra – 1
6	03.02 (Thu)	Part 1: Query Processing	Relational Algebra – 2
7	07.02 (Mon)	Part 1: Query Processing	Bag Relational Algebra
8	10.02 (Thu)	Part 1: Query Processing	Indexing – 1
9	14.02 (Mon)	Part 1: Query Processing	Indexing – 2
10	17.02 (Thu)	Part 1: Query Processing	Query compilation – 1
11	21.02 (Mon)	Part 1: Query Processing	Query compilation – 2
12	24.02 (Thu)	Part 1: Query Processing	Query evaluation – 1
13	28.02 (Mon)	Part 1: Query Processing	Query evaluation – 2

Mondays: 14:15–16:00, **Thursdays:** 12:15–14:00

First in *Zoom*, maybe physical later (I will keep you updated)

Lecture schedule – 2 (preliminary)

#	Date	Part	Topic
14	03.03 (Thu)	Part 2: TX Management	Overview and buffer management
15	07.03 (Mon)	Part 2: TX Management	ACID
16	10.03 (Thu)	Part 2: TX Management	Serialisation and concurrency – 1
17	14.03 (Mon)	Part 2: TX Management	Serialisation and concurrency – 2
18	17.03 (Thu)	Part 2: TX Management	Serialisation and concurrency – 3
19	21.03 (Mon)	Part 2: TX Management	Isolation levels – 1
20	24.03 (Thu)	Part 2: TX Management	Isolation levels – 2
21	28.03 (Mon)	Part 2: TX Management	Isolation levels – 3
22	31.03 (Thu)	Part 2: TX Management	Security in DBMS
23	04.04 (Mon)	for IN4020	Advanced DBMS
24	07.04 (Thu)	(spare)	(spare)
–	11.04 (Mon)	free	—
–	14.04 (Thu)	Easter break	—
–	18.04 (Mon)	Easter break	—

Mondays: 14:15–16:00, **Thursdays:** 12:15–14:00

First in *Zoom*, maybe physical later (I will keep you updated)

Lecture schedule – 3 (preliminary)

#	Date	Part	Topic
25	21.04 (Thu)	Part 3: NoSQL	Overview
26	25.04 (Mon)	Part 3: NoSQL	Column-based (MapReduce)
27	28.04 (Thu)	Part 3: NoSQL	Knowledge Graphs
28	02.05 (Mon)	Part 3: NoSQL	RDF and SPARQL
29	05.05 (Thu)	Part 3: NoSQL	Neo4j and Cypher
30	09.05 (Mon)	Emerging technologies	DBs and ML
31	12.05 (Thu)	Wrap-up	Summary

Mondays: 14:15–16:00, **Thursdays:** 12:15–14:00

First in **Zoom**, maybe physical later (I will keep you updated)

Preliminary: the length of these ‘lectures’ are often not yet tested, so we may **divert** from the time schedule

Main materials: Lecture slides and videos

- slides will be published (shortly) **before** the lecture at the Web page of the course:

<https://www.uio.no/studier/emner/matnat/ifi/IN3020/v22/timeplan/index.html>

- many slides are new, so may have typos: please, let me know (egork@ifi.uio.no) if you find bugs
- videos will be published (shortly) **after** the lecture at the same place

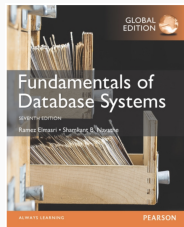
Secondary materials: Group and mandatory exercises (see below)

Other materials:

Fundamentals of Database Systems

by R. Elmasri and S.B. Navathe (7th ed.)

- only some **parts** are relevant (see below)
- **not everything** is covered
- many things are presented **differently**



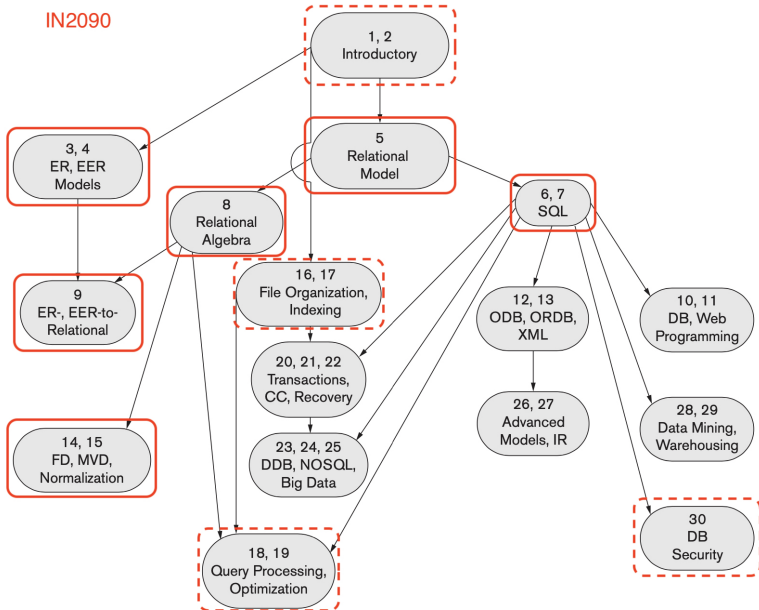
Other books and resources:

- *Foundations of Databases* by Abiteboul, Hull, and Vianu
<http://webdam.inria.fr/Alice/> (official)
- *Database systems – the complete book*
by Garcia-Molina, Ullman, and Widom (2nd ed.)
- **Wikipedia**, **PostgreSQL documentation**, etc.

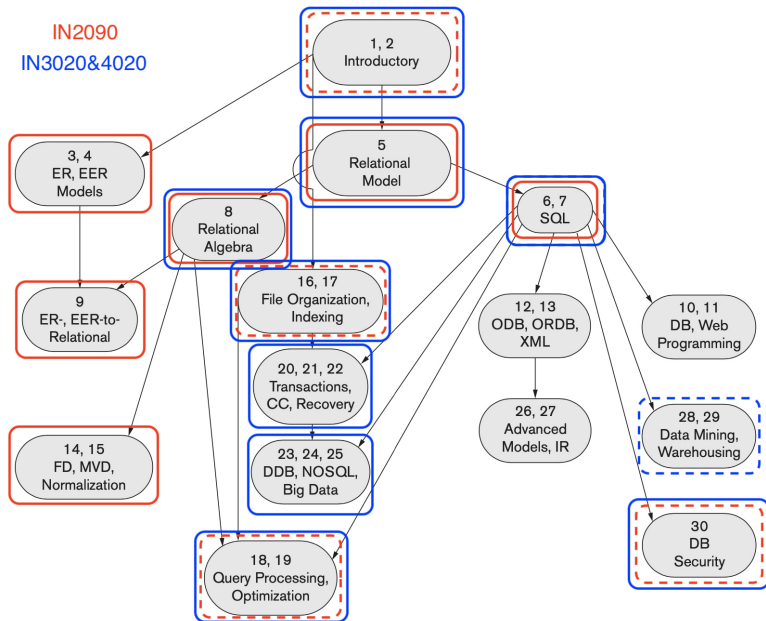
I will start **every lecture** with a list of relevant materials

Elmasri&Navathe chapter map (p. 13)

IN2090



Elmasri&Navathe chapter map (p. 13)



Group sessions

We have 2 or 3 **groups** with **weekly** sessions (classes)

- **Group 1:** Wed 14:15–16:00, Zoom/Seminarrom Prolog
Teacher: *Sondre Wold* (sondrewo@ifi.uio.no)
- **Group 2:** Mon 10:15–12:00, Zoom/Seminarrom Pascal
Teacher: *Ole Magnus Holter* (olemholt@ifi.uio.no)
- **Group 3(?):** Mon 10:15–12:00, Zoom only
Teacher: *Sondre Wold* (sondrewo@ifi.uio.no)

Each week except 17.01 (this), 11&18.04 (Easter), and maybe 21.04

Discuss **problems** (published on Thu on the Web page)

- it is expected that you have a look at them **in advance**
- roughly covers the **previous week** material (exception: the first class)
- One **more difficult problem** (with a 'star')

May include some demonstrations with **PostgreSQL** (see below)

Also discuss **completed mandatories** (see below)

Mandatory exercises

We have 4 **mandatories**:

#	Topic	Deadline
1	Relational Algebra	18.02 (Fri)
2	Query Processing	11.03 (Fri)
3	ACID and Concurrency	25.03 (Fri)
4	Isolation Levels	08.04 (Fri)

- Published **2 weeks** before the deadline
- Submitted in **Devilry**
- Check the **rules** for the mandatory exercises (obligement)
- All **necessary** for the exam
- **Extensions** are possible if properly justified (illness, etc.)
- Marked within two weeks (or quicker) by group teachers (**pass/not pass**)
- May be **resubmitted** if not passed
- Deadlines may **be changed** (will let you know in advance)
- **Short**

Exam formal details:

- **Time:** 30 May 2022, 3pm **Duration:** 4 hours
- **Place:** Silurveien 2 Sal 3D / Home **System:** Inspira
- **Withdrawal deadline:** 1 May

Exam rules and contents:

- no **collaboration** (standard plagiarism rules)
- formally, **no materials** are allowed during exam
- the content will depend on **the place**
 - if at home, slides will **not help** you much (so you will be allowed to use them)
- in any case, if you are comfortable with **mandatories**, you should be fine with the exam
- (**last-year exam** was a bit special, so do not look at it much)

Communication (we need extra effort)

Main entry point: the Web page of the course:

<https://www.uio.no/studier/emner/matnat/ifi/IN3020/v22/index.html>

- **schedule** of lectures and group sessions
- relevant **links** to Zoom meetings, slides, videos, group exercises
- **messages** (changes, mandatory sheets, links to other relevant systems)

Discussions: GitHub (no Mattemost, no Piazza, no Astroforum)

<https://github.uio.no/IN3020-4020/Discussions-2022>

- used **last year** (approved by UiO)
- has necessary **functionality** (can open and close discussion threads)
- all of you should already have **an account**, I will add all as users ASAP (let me know if you have problems accessing)
- we can create an **anonymous user** if needed (ask your friend to ask me)
- group teachers should **respond** within 1 working day (send me an email if you question is not answered)

Lectures: I stay for some time (Zoom/physical) after each lecture

Q1: Is this course theoretical or practical?

A1: **Both (or neither)**: no programming is involved, but we discuss how real software works

Q2: Do I need to know how to use SQL systems (PostgreSQL)?

A2: **Not necessary, but expected**: the mandatories and exam are on paper, but we will encourage you to experiment time to time and demonstrate how to do this

Q3: How different is the course from the last year?

A3: **Material is almost the same, presentation is (very) different**: this is the first time I teach this course as a leading lecturer, and I try to make it as I want it to be

- time-consuming, but I do my best
- lengths of lecture materials are not tested, so may change
- may be typos

Example DBMS architecture

