IN3020&4020 – Database Systems (2026)
Part: Intro & Recap
Lecture 1: Introduction and Motivation

---

19 January

Egor V. Kostylev
IFI, University of Oslo
egork@ifi.uio.no



About me

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**: Professor, University of Oslo

+ Neuro-Symbolic AI

About me

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**: Professor, University of Oslo

+ Neuro-Symbolic AI

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

About me

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**: Professor, University of Oslo

+ Neuro-Symbolic AI

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

How to find me: `egork@ifi.uio.no`, office 9461 in OJD

Lectures

Physical:

- main part of the course, usually 2 a week (other parts described below)
- an opportunity for a dialogue: interrupt me and ask questions
- I will ask questions sometimes, on the fly or using Menti

Lectures will be **recorded**:

- videos will be published (very) soon after the lecture
- as is and with no promise!

Lectures

Physical:

- main part of the course, usually 2 a week (other parts described below)
- an opportunity for a dialogue: interrupt me and ask questions
- I will ask questions sometimes, on the fly or using Menti

Lectures will be **recorded**:

- videos will be published (very) soon after the lecture
- as is and with no promise!
- do not forget about mandatories (see below):
  you need to follow the course during the term to succeed

What-this-is-about hierarchy

IN2090 – Databaser og datamodellering (Leif Harald Karlsen)

- what databases are and how to use them

IN3020&4020 – Database systems

- how databases work inside

IN5040 – Advanced database systems for Big Data
(Vera Hermine Goebel)

- new challenges in data management

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 expected, but ...)

IN3020&4020 – Database systems

- how databases work inside
- pre-requisite for IN5040

IN5040 – Advanced database systems for Big Data
(Vera Hermine Goebel)

- new challenges in data management

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 expected, but ...)

IN3020&4020 – Database systems

- how databases work inside
- pre-requisite for IN5040
- NOT an advanced SQL course!

IN5040 – Advanced database systems for Big Data
(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)

IN2090 – Databaser og datamodellering

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
   - syntax and semantics of Data Query Sub-Language
   - **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)

Database management system (DBMS):

- a software that facilitates defining, constructing, manipulating, and sharing databases by various users and applications

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

Database management system (DBMS):

- a software that facilitates defining, constructing, manipulating, and sharing databases by various users and applications
- *In this course*: SQL engines

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.

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?

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 that Python

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 that 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'

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 are 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, etc.

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 in the 'black box', so that we understand the common challenges and how to deal with them

Challenge 1: Query optimisation (example)

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

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

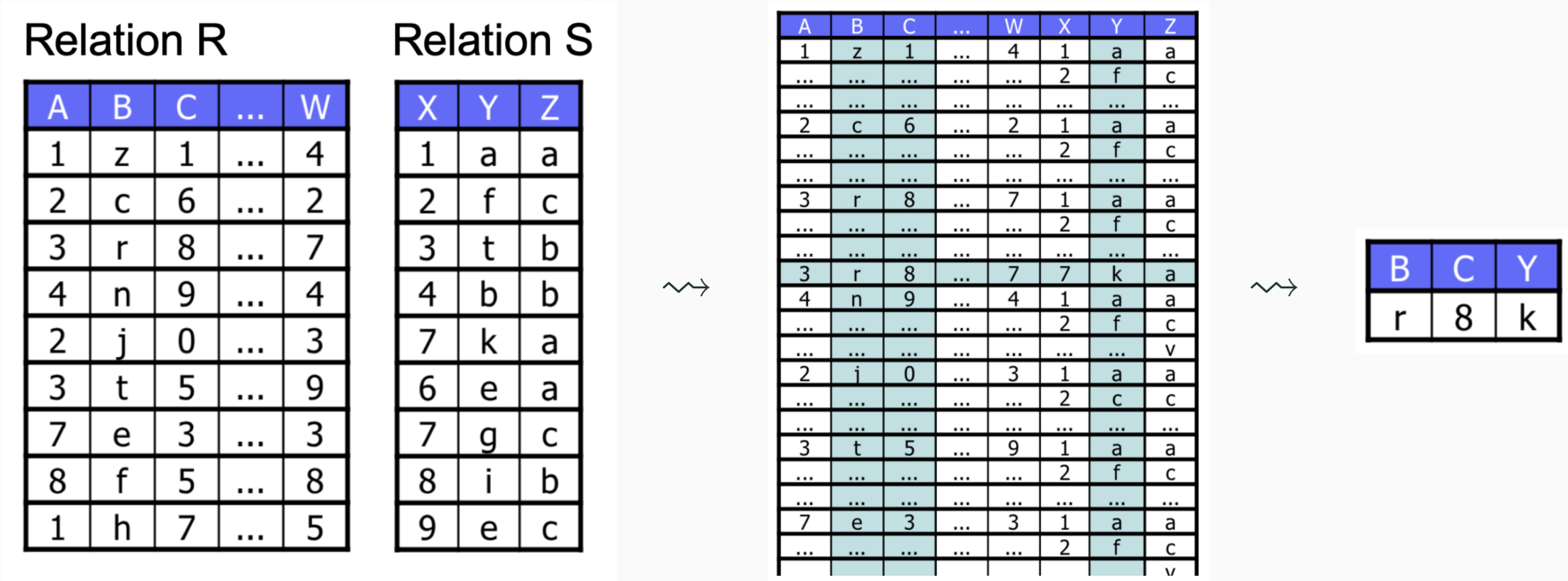
X	Y	Z
1	a	a
2	f	c
3	t	b
4	b	b
7	k	a
6	e	a
7	g	c
8	i	b
9	e	c

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 the 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

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

$\rightsquigarrow$

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	j	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

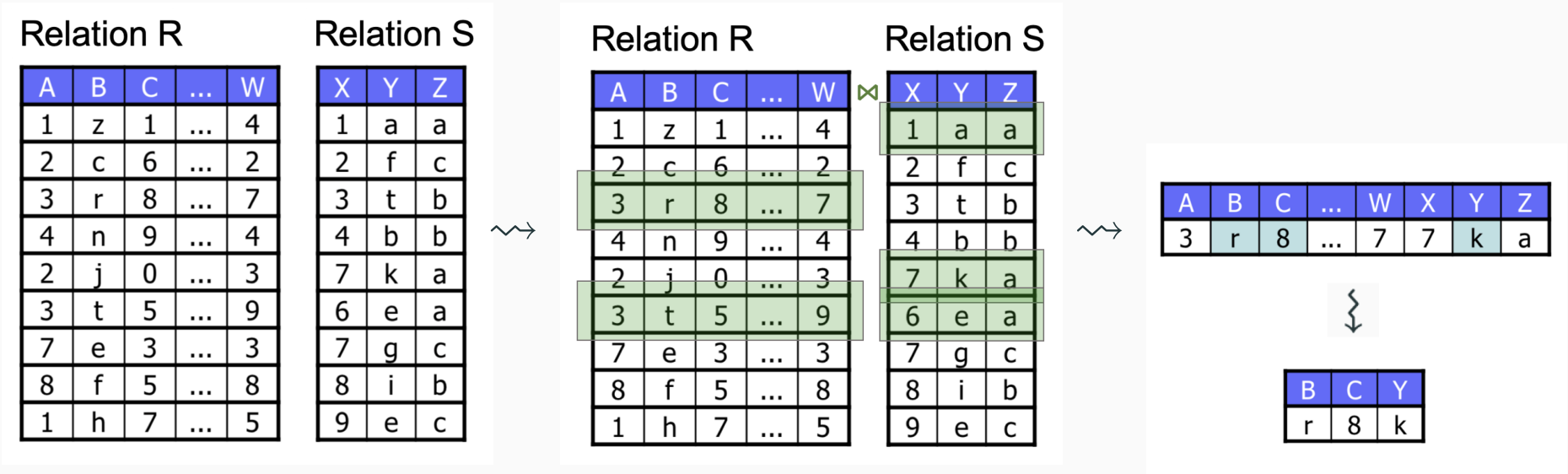
$\rightsquigarrow$

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 (much smaller tables to manipulate):
select relevant tuples in R and S; cross with selecting; project



Challenge 2: Efficient data storage (example)

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

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

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

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?

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

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`?

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

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 security

Syllabus

Three main parts:

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 (+ DB security)
3. NoSQL DBMS
   - knowledge graphs, column-based, etc.

Syllabus

Three main parts (different sizes!):

- 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 (+ DB security)

3. NoSQL DBMS
   - knowledge graphs, column-based, etc.

- Advanced DBMS Architecture (for IN4020)

Lecture schedule – 1 (preliminary, with rough borders)

#	Date	Part	Topic
1	19.01 (Mon)	Intro+SQL	Introduction
2	21.01 (Wed)	Intro+SQL	Relational Model
3	26.01 (Mon)	Intro+SQL	SQL recap – 1
4	28.01 (Wed)	Intro+SQL	SQL recap – 2
5	02.02 (Mon)	Part 1: Query Processing	Relational Algebra – 1
6	04.02 (Wed)	Part 1: Query Processing	Relational Algebra – 2
7	09.02 (Mon)	Part 1: Query Processing	Relational Algebra – 3
8	11.02 (Wed)	Part 1: Query Processing	Relational Algebra – 4
9	16.02 (Mon)	Part 1: Query Processing	Indexing – 1
10	18.02 (Wed)	Part 1: Query Processing	Indexing – 2
11	23.02 (Mon)	Part 1: Query Processing	Query compilation – 1
12	25.02 (Wed)	Part 1: Query Processing	Query compilation – 2
13	02.03 (Mon)	Part 1: Query Processing	Query evaluation – 1
14	04.03 (Wed)	Part 1: Query Processing	Query evaluation – 2

Mondays: 10:15–12:00 (OJD, Seminarrom Caml)

Wednesdays: 10:15–12:00 (OJD, Seminarrom Caml)

Lecture schedule – 2 (preliminary, with rough borders)

#	Date	Part	Topic
15	09.03 (Mon)	Part 2: TX Management	Transaction Concepts – 1
16	11.03 (Wed)	Part 2: TX Management	Transaction Concepts – 2
17	16.03 (Mon)	Part 2: TX Management	Transaction Concepts – 3
18	18.03 (Wed)	Part 2: TX Management	Concurrency Control – 1
19	23.03 (Mon)	Part 2: TX Management	Concurrency Control – 2
20	25.03 (Wed)	Part 2: TX Management	Concurrency Control – 3
21	08.04 (Wed)	Part 2: TX Management	Recovery Protocols – 1
22	13.04 (Mon)	Part 2: TX Management	Recovery Protocols – 2
23	15.04 (Wed)	(Part 2+)	Intro do DB Security
24	20.04 (Mon)	Part 3: NoSQL	NoSQL Overview
25	22.04 (Wed)	Part 3: NoSQL	Graph Databases – 1
26	27.04 (Mon)	Part 3: NoSQL	Graph Databases – 2
27	29.04 (Wed)	Part 3: NoSQL	Graph Databases – 3
28	04.05 (Mon)	(no part, for IN4020)	DBMS for Big Data
29	06.05 (Wed)	Summary and Exam	Summary
30	11.05 (Mon)	Summary and Exam	2025 Exam analysis

Mondays: 10:15–10:00, Wednesdays: 10:15–12:00

Materials

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/v26/timeplan/index.html`

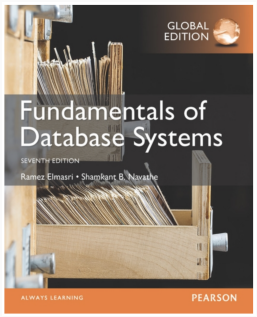
- slides may have typos: please, let me know (`egork@ifi.uio.no`) if you find bugs
- videos will be published (shortly) after the lectures at the page

Secondary materials: Group and mandatory exercises

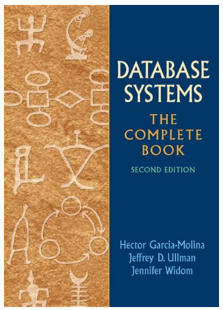
(+ 2022–25 exams, see below)

Materials

Other materials:



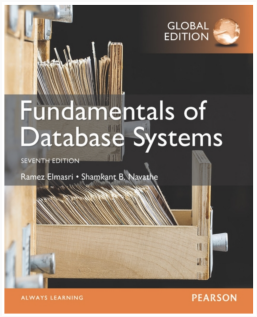
1. *Fundamentals of Database Systems*
   by R. Elmasri and S.B. Navathe (7th ed.)
2. *Database systems – the complete book*
   by H. Garcia-Molina, J.D. Ullman, and J. Widom (2nd ed.)
   - both: only some parts are relevant
   - both: not everything is covered
   - both: many things are presented differently



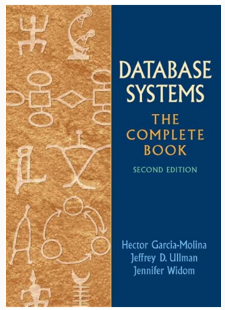
3. Other books and resources:
   - *Foundations of Databases* by Abiteboul, Hull, and Vianu
     `http://webdam.inria.fr/Alice/` (official)
   - Wikipedia, PostgreSQL documentation, etc.

Materials

Other materials:



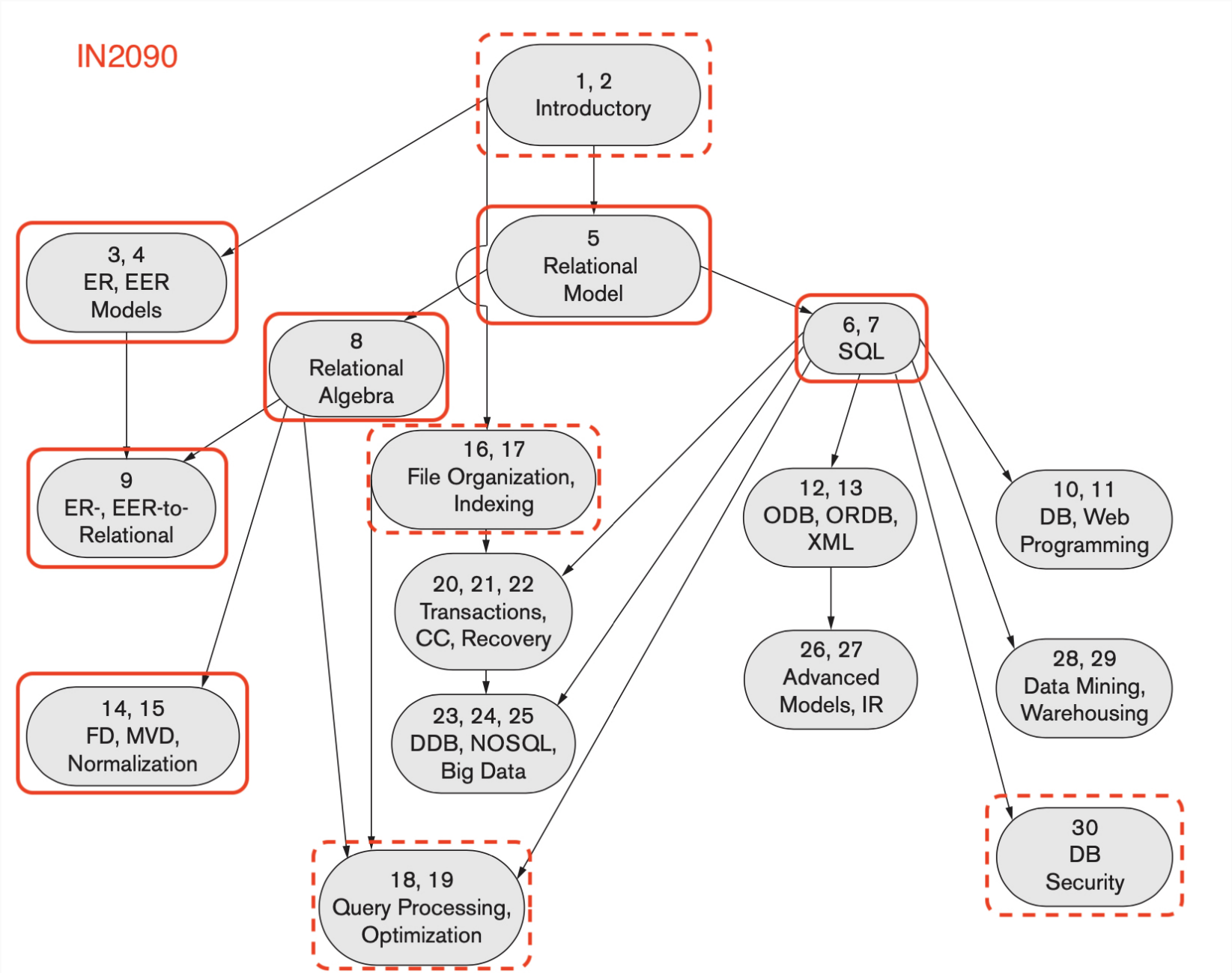
1. *Fundamentals of Database Systems*
   by R. Elmasri and S.B. Navathe (7th ed.)
2. *Database systems – the complete book*
   by H. Garcia-Molina, J.D. Ullman, and J. Widom (2nd ed.)
   - both: only some parts are relevant
   - both: not everything is covered
   - both: many things are presented differently



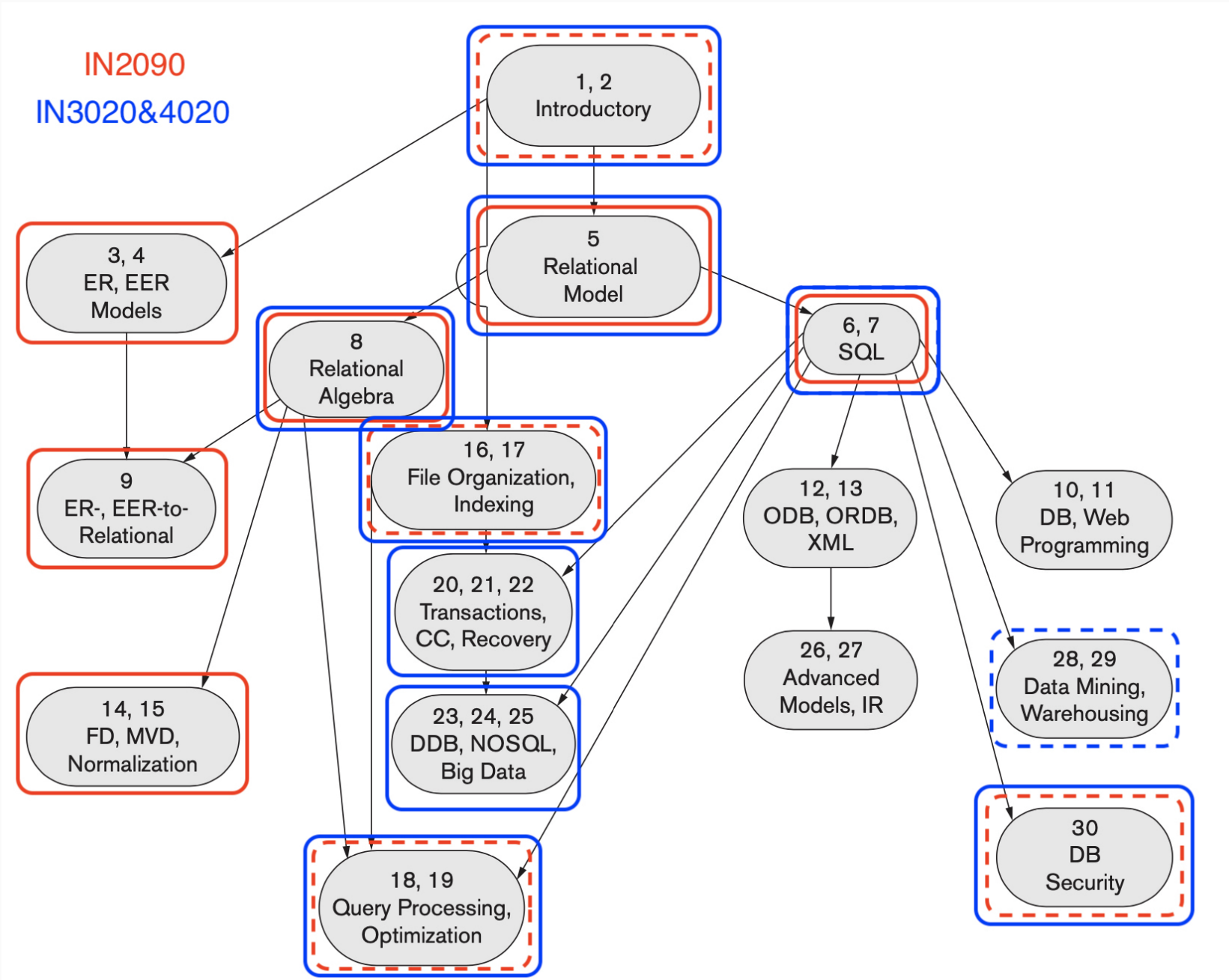
3. Other books and resources:
   - *Foundations of Databases* by Abiteboul, Hull, and Vianu
     `http://webdam.inria.fr/Alice/` (official)
   - Wikipedia, PostgreSQL documentation, etc.

I will start every lecture with a list of relevant materials

Elmasri&Navathe chapter map (p. 13)



Elmasri&Navathe chapter map (p. 13)



Group sessions

We have 2 groups with weekly sessions (each week except this)

- Group 1: Wed 12:15–14:00, Datastue Limbo
- Group 2: Tue 12:15–14:00, Datastue Limbo

Teacher (for both): *Sirin Koca* (`sirinkoc@ifi.uio.no`)

Discuss problems (published on previous Monday each week)

- it is desirable that you have a look at them in advance
- roughly covers the previous week material
- a more difficult problem (with a 'star', similar to mandatories and exam)
- solutions will be published (after the Friday's session)

May include some demonstrations with PostgreSQL (see below)

Also discuss completed mandatories (see below)

Mandatory exercises

We have 3 mandatories:

#	Topic	Deadline (preliminary)
1	Relational Algebra	20 Feb
2	Query Processing	13 Mar
3	Transaction Management	17 Apr

- Published 2 weeks before the deadline
- Submitted in Devilry
- Check the rules for the mandatory exercises (obligreglement)
- 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

Exam formal details:

- Time: 29 May 2025, 9am Duration: 4 hours
- Place: Silurveien 2 Sal 3C System: Inspera
- Withdrawal deadline: 13 May

Exam rules and contents:

- no collaboration (standard plagiarism rules)
- lecture slides are available during the exam
- no other materials are available
- you should be able to apply lecture materials
- if you are comfortable with mandatories, you should be fine with the exam
- 22–25 exams (but not the before) is good training material
  (do not recommend to look at 25 before the final training)

Communication

Main entry point: the Web page of the course:

`https://www.uio.no/studier/emner/matnat/ifi/IN3020/v26/index.html`

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

Discussions: Astro Discourse

`https://discourse.uio.no/c/in3020-in4020-26v/849` (updated link!)

- approved by UiO, have good reputation from other lecturers
- student instructions can be found on Web page of the course
- We should respond within 1 working day

Lectures: I stay for some time after each lecture

FAQ

Q1: Is this course theoretical or practical?

FAQ

Q1: Is this course theoretical or practical?

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

FAQ

Q1: Is this course theoretical or practical?

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

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

FAQ

Q1: Is this course theoretical or practical?

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

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

FAQ

Q1: Is this course theoretical or practical?

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

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?

FAQ

Q1: Is this course theoretical or practical?

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

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: Quite similar: you may look at slides and videos, but nothing is promised (I may change and adapt some parts)

Example DBMS architecture

