NTNU
Kunnskap for en bedre verden

CANDIDATE
10172

TEST

TDT4145 1 Datamodellering og databasesystemer

Subject code	TDT4145
Evaluation type	Skriftlig eksamen
Test opening time	24.05.2019 07:00
End time	24.05.2019 11:00
Grade deadline	14.06.2019 21:59
PDF created	05.08.2020 11:46

Seksjon 1

Question	Question title	Question type
i	Front page	Document
1	Problem 1 (15 %)	Oral
2	Problem 2 (4 %)	Oral
3	Problem 3 (4 %)	Essay
4	Problem 4 (4 %)	Essay
5	Problem 5 (4 %)	Essay
6	Problem 6 (8 %)	Essay
7	Problem 7 (2 %)	Essay
8	Problem 8 (2 %)	Essay
9	Problem 9 (5 %)	Essay
10	Problem 10 (5 %)	Essay
11	Problem 11 (3 %)	Essay
12	Problem 12 (4 %)	Oral
13	Problem 13 (5 %)	Essay
14	Problem 14 (2.5 %)	Essay
15	Problem 15 (2.5 %)	Essay
16	Problem 16 (2.5 %)	Essay
17	Problem 17 (2.5 %)	Essay
18	Problem 18 (5 %)	Essay
19	Problem 19 (5 %)	Essay
20	Problem 20 (5 %)	Essay
21	Problem 21 (5 %)	Essay

22 Problem 22 (5 %) Essay

1

Problem 1 (15 %)

Make an Entity-Relationship diagram for the following case, a simplified version of a study programme database. You may use all ER modelling concepts from the curriculum.

Study programmes have a unique programme code, a study programme name, and span a number of semesters. There are several types of study programmes: One-year programmes (“årsstudier”) over two semesters, three-year Bachelor programmes over six semesters, two-year master’s programmes over four semesters, and five-year master’s programmes over 10 semesters. Within a study programme, semesters are numbered one, two, and so on. All semesters have a semester code, which designates the semester as a spring semester or a fall semester.

Each semester in a programme has a number of courses (“emner”), each having a unique course code, a course name and a study point credit. A course can be part of many study programmes. A course can be mandatory or optional within a study programme. Each course has a semester code that shows whether it is a fall semester course, a spring semester course or both. It is possible to have courses, which are not part of any study programme.

Learning outcome descriptions states what a person know, can do and is capable of doing as a result of a learning process. Learning outcome descriptions are formulated at two levels; program level learning outcome descriptions and learning outcome descriptions for a single course. A learning outcome description is classified as either “knowledge”, “skills” or “general compentences”. Every study programme will have a number of overall learning outcome descriptions, and every course will have a number of (course level) learning outcome descriptions. A learning outcome description, at any level, will have a unique learning outcome description identifier (LUB-ID), a defining text, and a field that shows whether this is “knowledge”, “skills” or “general compentences”. A learning outcome description at program level will be associated with one specific study programme. A course level learning outcome description will be for one course only.

We want to know how course level learning outcome descriptions contribute to fulfil program level learning outcome descriptions. To be able to track that, we must be able to register that a course level learning outcome description contributes to the fulfilment of one or more programme level learning outcome descriptions.

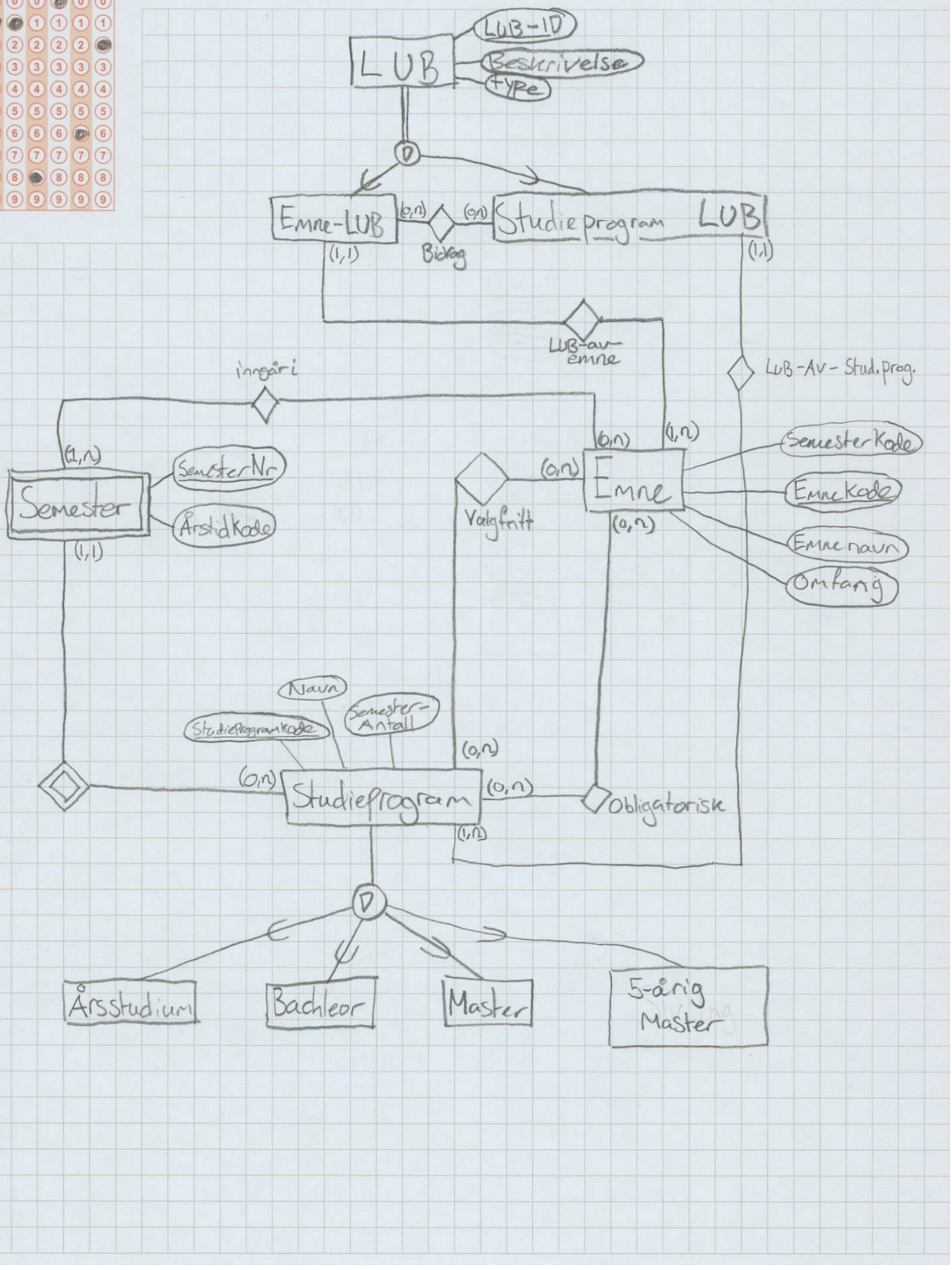
State any assumptions you find necessary.

This problem must be solved on paper.

Attaching sketches to this question?
Use the following code:

4 1 1 8 0 6 2

Question Code Oppgavekode	Date Dato	Subject code Emnekode	Candidate ID KandidatID	Question nr Oppgavenr	Page number Sidetall
4 1 1 8 0 6 2	24.5	TDT4145	10172	1	1



Question Code Oppgavekode	Date Dato	Subject code Emnekode	Candidate ID KandidatID	Question nr Oppgavenr	Page number Sidetall
4 1 1 8 0 6 2	24.05	TDT4145	10172	1	2

Antagelser ved ER-Diagram:

- Siden det står at et studieprogram "kan" være en av de 4 typene, tolket jeg dette som at spesialiseringen ikke var total. Dette stemmer overens med at det finnes også andre typer studieprogram ikke nevnt her.
- Semester må ha minst ett emne for å være et semester.
- LUB gjelder kun ett emne eller ett studieprogram. ~~Ingen~~ Ingen to emner skal gir like læringsutbytte, heller ingen to studieprogram.
- Emne haren egen Unik Emnekode F.eks "TDT4145"
- Et Studieprogram og et emne må ha minst én LUB.
- LUB sin "Type" Attributt betegner om det er kunnskap, ferdighet osv.

Merknad: Jeg vurderte å ha betingelse på arven mellom Studieprogram og Sine Subklasser, men ettersom det finnes enkelte profesjonsstudier Som kar 10 semestre ville ikke dette gått.

2 Problem 2 (4 %)

Use the following relational database (primary keys are underlined):

Course(CourseCode, CourseTitle, CourseCredit, CourseDecription)

Student(StudentNo, FirstName, LastName, Gender)

Exam(ID, ExamDay, ExamMonth, ExamYear, CourseCode, Duration)

CourseCode is foreign key referencing Course, cannot take NULL value

ExamResults(ExamID, StudentNo, Grade)

ExamID is foreign key referencing Exam, cannot take NULL value

StudentNo is foreign key referencing Student, cannot take NULL value

Write a query in relational algebra to find StudentNo, FirstName, Lastname, ExamYear and CourseCode for all students given the grade «A» in courses with course title «Datamodellering og databasesystemer».

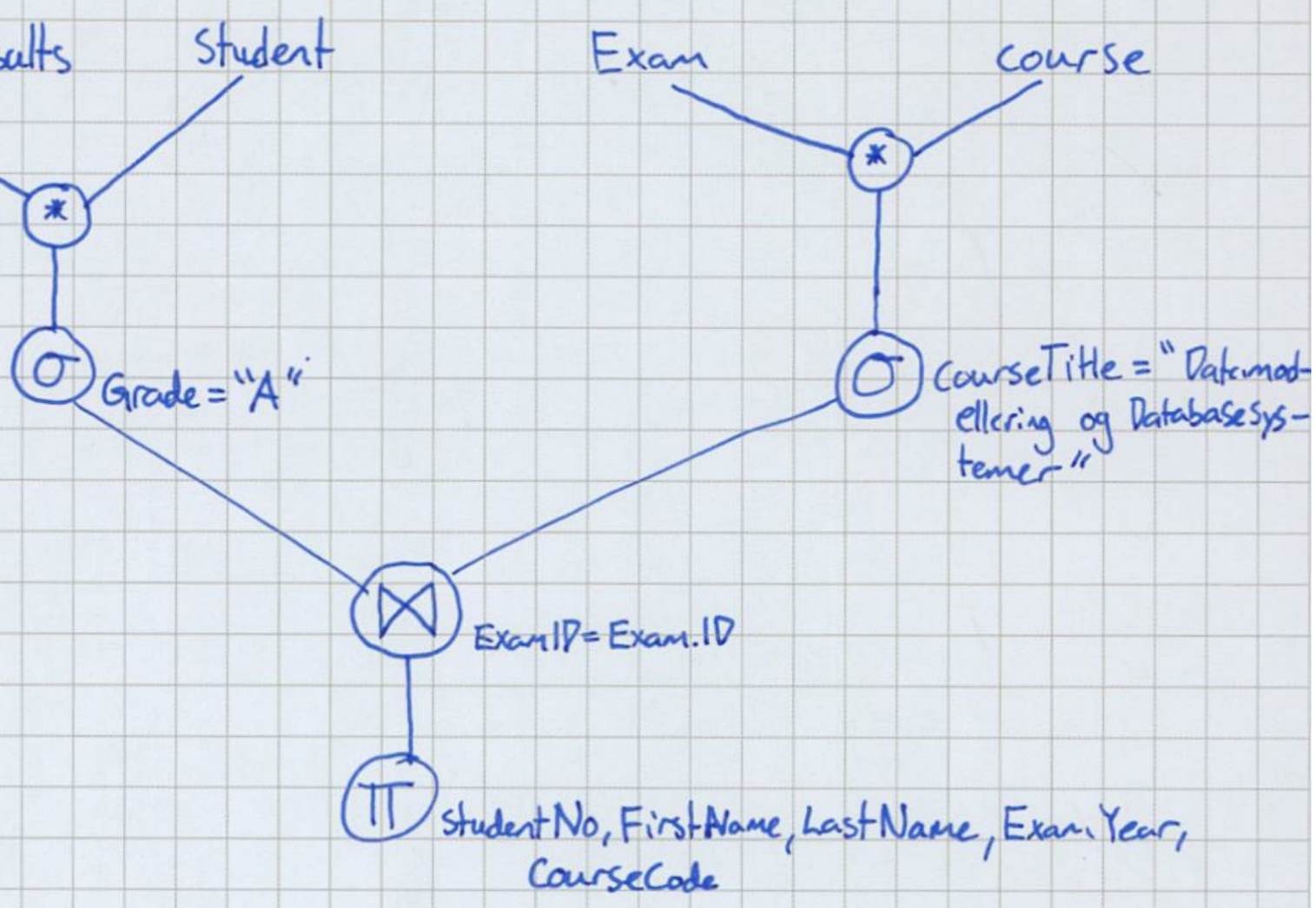
Relational algebra can be stated as text or graphs. If you master both notations, we encourage you to formulate your answer in the graphical form. However, a correct query stated as text will get full score.

This problem must be solved on paper.

Attaching sketches to this question?
Use the following code:

4 1 9 8 9 5 3

Question Code Oppgavekode	Date Dato	Subject code Emnekode	Candidate ID KandidatID	Question nr Oppgavenr	Page number Sidetall
4 1 9 8 9 5 3	24.5	TDT4145	10172	2	3



3

Problem 3 (4 %)

Use the following relational database (primary keys are underlined):

Course(CourseCode, CourseTitle, CourseCredit, CourseDecription)

Student(StudentNo, FirstName, LastName, Gender)

Exam(ID, ExamDay, ExamMonth, ExamYear, CourseCode, Duration)

CourseCode is foreign key referencing Course, cannot take NULL value

ExamResults(ExamID, StudentNo, Grade)

ExamID is foreign key referencing Exam, cannot take NULL value

StudentNo is foreign key referencing Student, cannot take NULL value

Write SQL to find the number of exam results in TDT4142 (CourseCode) for each year there has been an exam in that course. The result should be sorted so that data from the most recent years come first.

Fill in your answer here

```
SELECT ExamYear, Count(ExamID, StudentNo)

FROM Exam INNER JOIN ExamResults ON (ExamID = ID)

WHERE CourseCode = "TDT4145"

GROUP BY ExamYear

ORDER BY ExamYear DESC
```

Attaching sketches to this question?

Use the following code:

0 6 2 9 3 8 6

4 Problem 4 (4 %)

Use the following relational database (primary keys are underlined):

Course(<u>CourseCode</u>, CourseTitle, CourseCredit, CourseDecription)

Student(<u>StudentNo</u>, FirstName, LastName, Gender)

Exam(<u>ID</u>, ExamDay, ExamMonth, ExamYear, CourseCode, Duration)

CourseCode is foreign key referencing Course, cannot take NULL value

ExamResults(<u>ExamID, StudentNo</u>, Grade)

ExamID is foreign key referencing Exam, cannot take NULL value

StudentNo is foreign key referencing Student, cannot take NULL value

Write SQL to find CourseCode and CourseTitle for all courses which have had at least one exam and where no student has achieved an “A” so far.

Fill in your answer here

```
SELECT CourseCode, CourseTitle

FROM Course

WHERE CourseCode NOT IN (

SELECT CourseCode

FROM ExamResults

WHERE Grade = "A"

)

AND CourseCode IN (

SELECT DISTINCT CourseCode

FROM Exam

)
```

Attaching sketches to this question?
Use the following code:

3 8 0 3 2 4 9

5 # Problem 5 (4 %)

Use the following relational database (primary keys are underlined):

Course(<u>CourseCode</u>, CourseTitle, CourseCredit, CourseDecription)

Student(<u>StudentNo</u>, FirstName, LastName, Gender)

Exam(<u>ID</u>, ExamDay, ExamMonth, ExamYear, CourseCode, Duration)

CourseCode is foreign key referencing Course, cannot take NULL value

ExamResults(<u>ExamID, StudentNo</u>, Grade)

ExamID is foreign key referencing Exam, cannot take NULL value

StudentNo is foreign key referencing Student, cannot take NULL value

Write SQL that creates the ExamResults table. Choose appropriate data types and state any assumptions you find necessary.

Fill in your answer here

CREATE TABLE ExamResults(ExamID NOT NULL INT(64) FK(Exam) ,

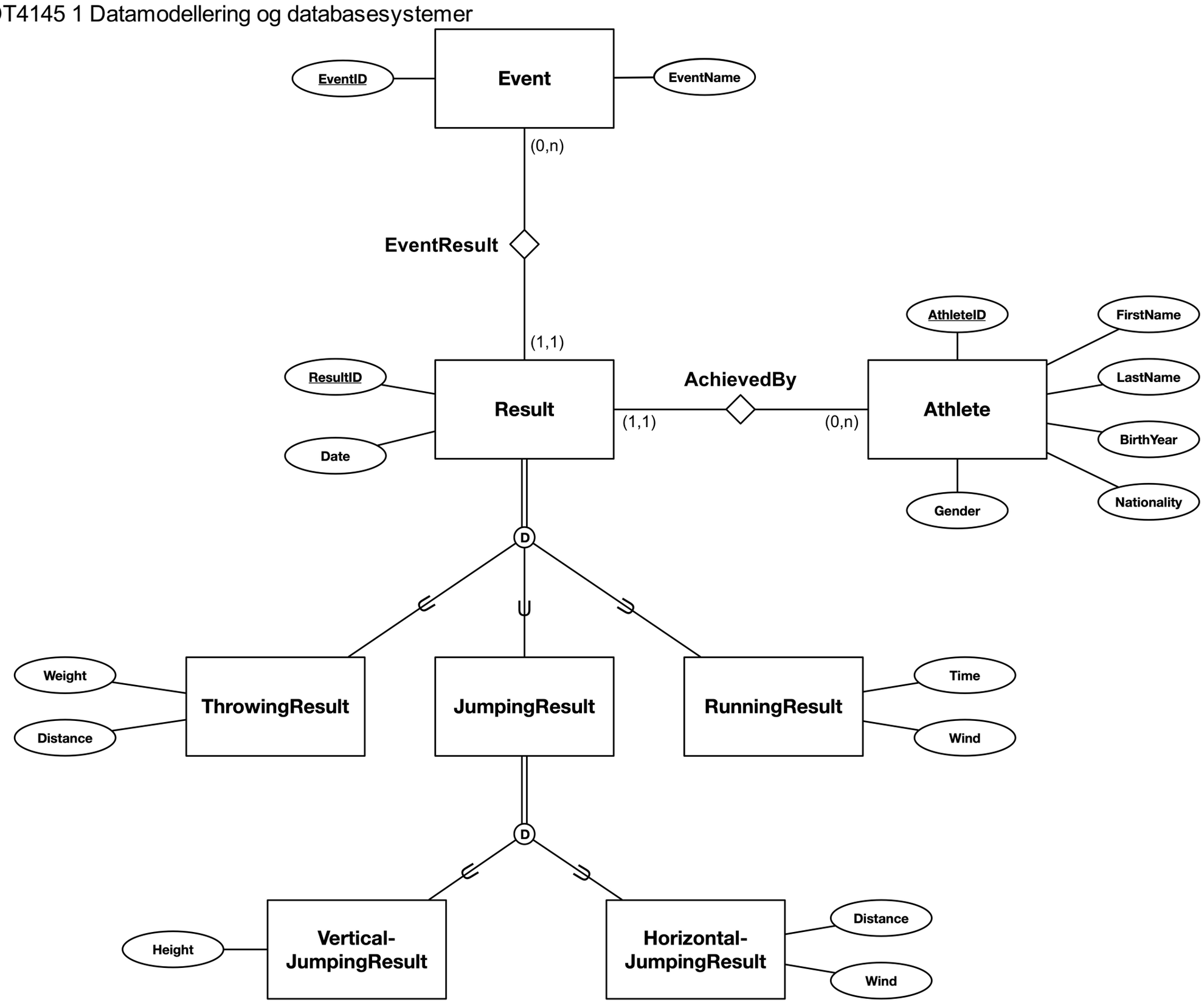
StudentNo NOT NULL INT(64) FK(StudentNo), Grade CHAR(1))

Attaching sketches to this question?
Use the following code:

0 6 7 4 2 0 8

6 # Problem 6 (8 %)

In the diagram below, we have shown an Entity-Relationship model designed to keep track of results in a slightly simplified version of Track & Fields.



Map this Entity-Relationship model into a corresponding relational database schema (model).
Explain why your proposed schema is better that the possible alternatives.

State any assumptions you find necessary.

Fill in your answer here

Event(<u>EventID</u>, EventName)

Athlete(AthleteID, FirstName, LastName, BirthYear, Nationality, Gender)

ThrowingResult(Weight, Distance, ResultID, Date, EventID, AthleteID)

- EventID er frmmednøkkel til Event
- AthleteID er fremmednøkkel til Athlete

RunningResult(Time, Wind, ResultID, Date, EventID, AthleteID)

- EventID er frmmednøkkel til Event
- AthleteID er fremmednøkkel til Athlete

VerticalJumpingResult(Height, ResultID, Date, EventID, AthleteID)

- EventID er frmmednøkkel til Event
- AthleteID er fremmednøkkel til Athlete

HorizontalJumpingResult(Distance, Wind, ResultID, Date, EventID, AthleteID)

- EventID er frmmednøkkel til Event
- AthleteID er fremmednøkkel til Athlete

Denne løsningen er gunstigere enn alternativet av å lagre alle attributtene i superklassen, da spesialiseringen er disjunkt. Dette påfører at lagring av alle subklasser i en stor tabell av superklassen er sløsende, da kun maksimalt 2 av 7 attributtene påført av subklassene vil være NON-NULL samtidig. Denne løsningen fører derimot til flere join'er i enkelte tilfeller, men ettersom det ikke er gitt hva som er en typisk spørring mot denne databasen anser jeg dette som mindre viktig enn sløsing av lagringsplass. Dette gjelder forsåvidt også for grunnlaget til å bruke horizontal og vertical hver for seg.

Attaching sketches to this question?
Use the following code:

0 4 8 5 6 8 7

7 **Problem 7 (2 %)**

A relational table (relation) will always have at least one *key*. Explain why this is the case.

Fill in your answer here

En tabell må ha minst en attributt for å være en tabell. En nøkkel er en samling av attributter(i dette tilfellet ett) i tabellen. En attributt er i seg selv en nøkkel, selv om den ikke nødvendigvis unikt identifiserer radene. Derfor vil en tabell alltid ha en nøkkel.

Attaching sketches to this question?
Use the following code:

9 3 1 8 6 0 5

8 **Problem 8 (2 %)**

Explain the differences between a *key* and a *superkey*.

Fill in your answer here

A superkey is a key which is a unique identifier for the table. A superkey is a type of key. Whereas a key **might** not uniquely identify rows in the table, a superkey must do so to be a superkey. Another key is a candidate key which is a minimal superkey.

Attaching sketches to this question?
Use the following code:

5 2 3 9 6 2 4

9

Problem 9 (5 %)

Assume a relational table is decomposed (split) into two component tables. This split may not preserve the functional dependencies. Explain what this means. Make an example of a decomposition of this kind and discuss the consequences of not preserving the functional dependencies.

Fill in your answer here

Hvis en dekomponering ikke har bevaring av funksjonelle avhengigheter vil $F = F_1 \cup F_2 \cup \ldots \cup F_n$ ikke gjelde. Dette betyr at settet av totalt de funksjonelle avhengigheter av tabellene i dekomponeringen er ikke likt til settet av funskjonelle avhengigheter som gjaldt før dekomponeringen.

Hvis vi starter med følgende tabell:

R(A, B, C, D)

F{A->C}

og dekomponerer slik:

R_1(A, B) , R_2(C, D)

Da vil den ene funksjonelle avhengigheten vi hadde i R ikke gjelde i noen av tabellene i dekomponeringen. Vi mister dermed den ene funksjonelle avhengigheten vi hadde og funksjonsbevaring er ikke opprettholdt.

Attaching sketches to this question?
Use the following code:

2 8 4 1 6 9 4

10

Problem 10 (5 %)

Base your answer on the assumption that all relevant constraints can be stated as functional dependencies. Consider a table that satisfies first normal form (1NF) but does not satisfy second normal form (2NF).

Explain the problems that can occur as a result of this table not satisfying Boyce-Codd normal form (BCNF). The (root) causes of these problems can be classified into three classes. Describe these three classes

Fill in your answer here

Innsettingsproblemer: En instans av dekomponeringstabellene kan ikke eksistere før det eksisterer en instans av den i den udekomponerte tabellen. F.eks en kunde av et selskap kan ikke eksistere i databasen uten å ha en ordre liggende inn i databasen. Innsetting av rader er også mye mer tungvindt da dette krever å føre informasjon på attributter som kan utlede dette på egenhånd. F.eks en ordre må logge kundens telefonnummer hver gang selv om dette kunne utledes fra kundens ID.

Endringsproblemer: En endring av ikke-nøkkel attributt vil påkreve oppdatering av alle rader som har samme verdi i feltet denne attributten er funksjonelt avhengig av. F.eks ending av telefonnummer til en kunde vil kreve oppdatering av alle ordre gjort av denne kunden.

Slettingsproblemer: Ved sletting av rader kan informasjon om andre ting gå tapt. F.eks ved sletting av en ordre kan informasjon om produktet og kundene forsvinne om det ikke finnes en annen rad i tabellen med denne informasjonen.

Attaching sketches to this question?

Use the following code:

0 3 5 6 7 5 6

11 Problem 11 (3 %)

In which cases do we use categories in an EER model?

Fill in your answer here

Kategorier er hensiktsmessig å bruke når vi har en tabell som kan ta formen av sine kategorier. Den arver da attributtene som kategorien den tar formen som. Vi bruker kategorier når vi skal snakke om et fellesobjekt som kan ta flere former og bruksområder, og den "har" attributter fra kategorien som den inntar.

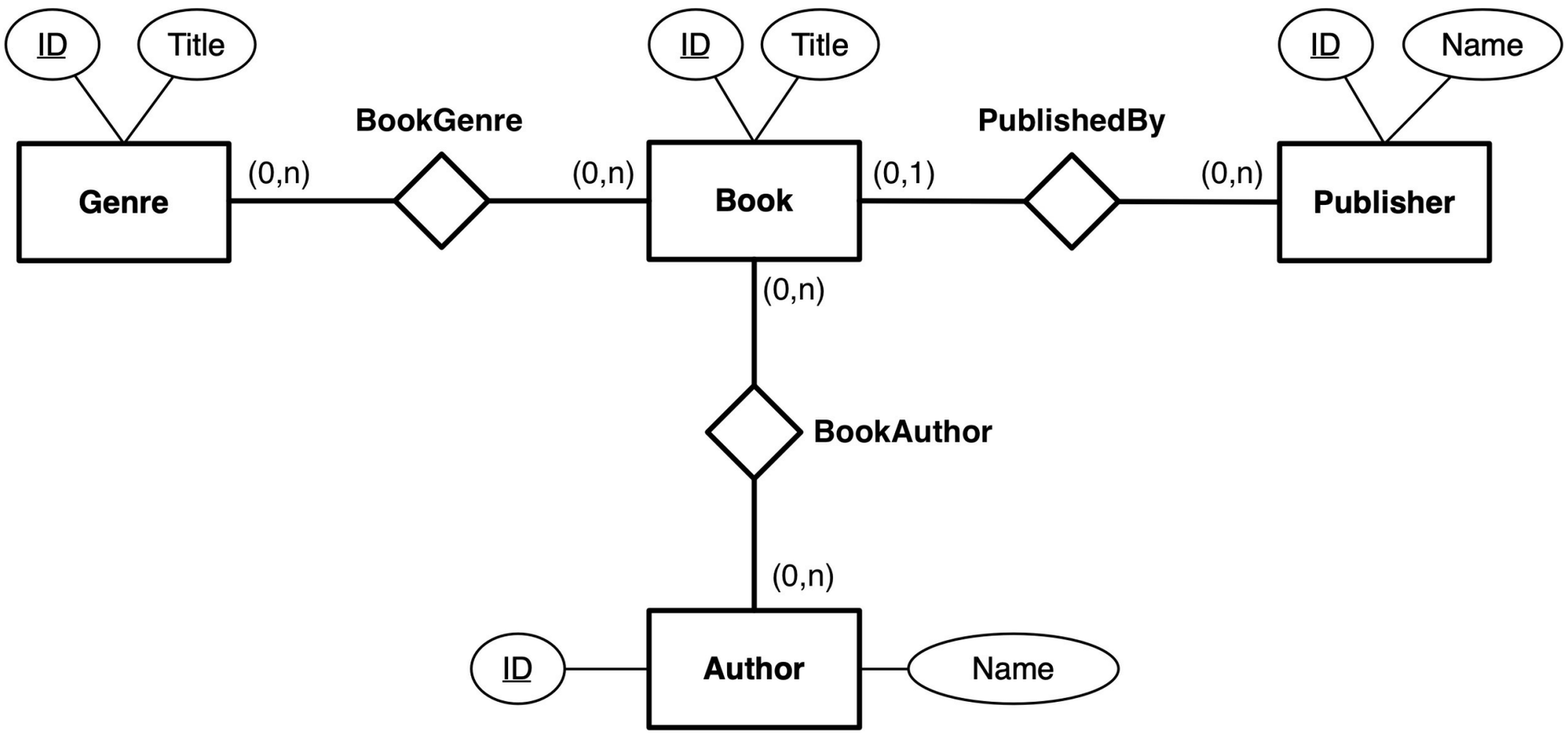
F.eks et bildetakningsapparat kan ta formen av et kamera eller en smarttelefon. Om apparatet tar formen av en smarttelefon vil det hente alle attributtene derfra slik som telefonnummer, samt ha sine egne attributter som ApparatID og bildekvalitet.

Attaching sketches to this question?
Use the following code:

2 8 3 3 2 3 8

12 Problem 12 (4 %)

In the diagram below we have shown an ER model for a simple book database



The tables below show data from a corresponding relational database.

Book

BookID	Title	PublisherID
1	Sult	1
2	Victoria	1
3	Factfulness	2
4	Thinking, Fast and Slow	3

Author

AuthorID	Name
1	Knut Hamsun
2	Hans Rosling
3	Daniel Kahneman

Publisher

PublisherID	Name
1	Gyldendal
2	Cappelen
3	Penguin
4	Oktober

Genre

GenreID	Title
1	Poetry
2	Novel
3	Nonfiction

BookGenre

BookID	GenreID
1	2
2	2
3	3
4	3

BookAuthor

BookID	AuthorID
1	1
2	1
3	2
4	3

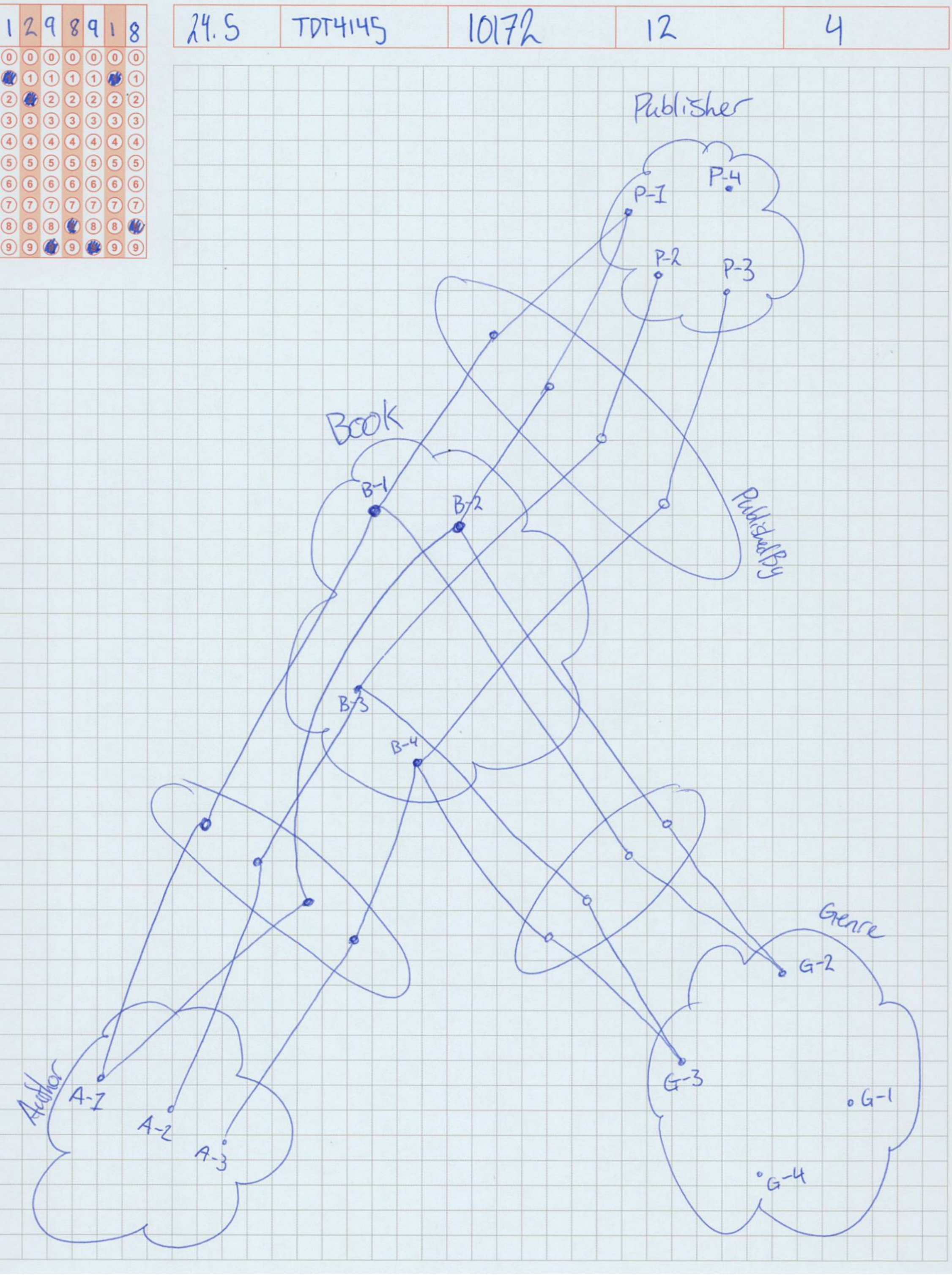
Draw an instance diagram («forekomstdiagram») for the ER model, based on the data from the relational database.

This problem must be solved on paper.

Attaching sketches to this question?
Use the following code:

1 2 9 8 9 1 8

Question Code Oppgavekode	Date Dato	Subject code Emnekode	Candidate ID KandidatID	Question nr Oppgavenr	Page number Sidetall
129891 8	24.5	TDT4145	10172	12	4



Question Code Oppgavekode	Date Dato	Subject code Emnekode	Candidate ID KandidatID	Question nr Oppgavenr	Page number Sidetall
1291918	24.5	TDT4145	10172	12	5

Labelene i forekomstdiagrammet er på formatet:

{første bokstav av tabellen} – {ID til objektet i sin tabell}

- Gjorde det slik fordi jeg hadde dårlig tid og rakk ikke føre inn alle navn og sånt.

13 # Problem 13 (5 %)

We have an *extendible hashing* data stucture. At start we had *global depth* 2. Now, the *global depth* is 5. Which values may *local depth* be in this extendible hashing data structure? Explain your answer.

Fill in your answer here

Vi kan ha lokal dybde på 2,3,4 og 5. Vi må derimot ha minst 2 blokker med $lokaldybde = globaldybe = 5$, ettersom for å oppnå globaldybde 5 må vi ha en blokk på $lokaldybe = globaldybde = 4$ som fyller seg opp og krever høyere lokaldybde, og dermed også høyere global dybde. Vi kan også ha blokker med $lokaldybde < 5$, da det kan være en blokk som ikke fylles opp like mye som den som pusher æøkning av globaldybde. F.eks. Hvis vi får bare oddetall og vi bruker hashfunksjonen $h(k) = k \, mod \, 2^5$ og start global dybde på 2, vil vi få to blokker med lokal dybde 2 selv om de to andre blokkene har høyere lokal dybde.

Attaching sketches to this question?
Use the following code:

2 8 9 7 7 2 6

14 # Problem 14 (2.5 %)

We have a table

Student (pno, studno, lastname, firstname, email)

The table is stored as a heapfile. We have a static hash index on **pno**. In addition we have got an unclustered B+-tree on **lastname.** Each record in the hashindex is on the format **(pno, RecordID)** and each leaf-level record in the B+-tree is on the format **(lastname, RecordID, RecordID, ...),** i.e., a variable number of RecordIDs per lastname.

The number of students in the table is 50 000. The heap file contains 1000 blocks. The B+-tree contains 500 blocks on the leaf level and it has 3 levels. The hash index contains 300 blocks and 60 overflow blocks and the average number of blocks accessed per lookup is 1.2.

Give an estimate of how many blocks are accessed with the following SQL statement. Explain your answer.

SELECT * FROM Student;

Fill in your answer here

Her kjører vi rett på heapfilen og scanner hele tabellen. Dette vil gi 1000 blokkaksesser.

Attaching sketches to this question?
Use the following code:

3 8 5 1 1 0 7

15

Problem 15 (2.5 %)

We have a table

Student (pno, studno, lastname, firstname, email)

The table is stored as a heapfile. We have a static hash index on **pno**. In addition we have got an unclustered B+-tree on **lastname**. Each record in the hashindex is on the format **(pno, RecordID)** and each leaf-level record in the B+-tree is on the format **(lastname, RecordID, RecordID, ...)**, i.e., a variable number of RecordIDs per lastname.

The number of students in the table is 50 000. The heap file contains 1000 blocks. The B+-tree contains 500 blocks on the leaf level and it has 3 levels. The hash index contains 300 blocks and 60 overflow blocks and the average number of blocks accessed per lookup is 1.2.

Give an estimate of how many blocks are accessed with the following SQL statement. Explain your answer.

SELECT * FROM Student WHERE pno=12121212345;

Fill in your answer here

Direkte aksess på hashstrukturen: 1.2 blokk,

blokk i heapfila som man følger til fra hashstrukturen: 1 blokk,

Totalt: 2.2 blokkaksesser i snitt.

Attaching sketches to this question?
Use the following code:

9 2 8 7 0 3 9

16

Problem 16 (2.5 %)

We have a table

Student (pno, studno, lastname, firstname, email)

The table is stored as a heapfile. We have a static hash index on **pno**. In addition we have got an unclustered B+-tree on **lastname**. Each record in the hashindex is on the format **(pno, RecordID)** and each leaf-level record in the B+-tree is on the format **(lastname, RecordID, RecordID, ...),** i.e., a variable number of RecordIDs per lastname.

The number of students in the table is 50 000. The heap file contains 1000 blocks. The B+-tree contains 500 blocks on the leaf level and it has 3 levels. The hash index contains 300 blocks and 60 overflow blocks and the average number of blocks accessed per lookup is 1.2.

Give an estimate of how many blocks are accessed with the following SQL statement. Explain your answer.

SELECT * FROM Student WHERE lastname='Hansen';

Fill in your answer here

Traversering av B+ treet: 3 blokker

For hver RecordID i posten skal vi slå opp heapfilen. Dette kan gi oss $1\% * 50000 = 500$blokkaksesser da vi ikke hvet hvor i heapfilen disse er lagret.

Dette gir oss en total blokkaksess på 503.

Antar: lastname er unikt lagret i B+ treet der alle med samme LastName har sine recordID samlet i en post. Antar at 1% av stundentene heter Hansen til etternavn. Hadde det vært en større andel som het hansen kunne det vært hensiktsmessig å kun bruke heapfila og få en grense på 1000 blokkaksesser. Men i dette tilfellet av 1% er det best å bruke B+ treets pekere til Heapfilen.

Attaching sketches to this question?
Use the following code:

2 0 1 2 3 6 9

17

Problem 17 (2.5 %)

We have a table

Student (pno, studno, lastname, firstname, email)

The table is stored as a heapfile. We have a static hash index on **pno**. In addition we have got an unclustered B+-tree on **lastname**. Each record in the hashindex is on the format **(pno, RecordID) a**nd each leaf-level record in the B+-tree is on the format **(lastname, RecordID, RecordID, ...),** i.e., a variable number of RecordIDs per lastname.

The number of students in the table is 50 000. The heap file contains 1000 blocks. The B+-tree contains 500 blocks on the leaf level and it has 3 levels. The hash index contains 300 blocks and 60 overflow blocks and the average number of blocks accessed per lookup is 1.2.

Give an estimate of how many blocks are accessed with the following SQL statement. Explain your answer.

SELECT DISTINCT lastname FROM Student;

Fill in your answer here

Detter er et index-only-query, så vi kan traversere ned til løvnivået i B+ treet og dermed traversere bortover. Bortover i dette treet er lastname lagret som index, og de er distincte fra før av så her trenger vi ingenting annet enn å gå innom hver post bortover. Dette gir 2+500 blokkaksesser = 502.

Attaching sketches to this question?
Use the following code:

9 4 8 8 6 2 5

18

Problem 18 (5 %)

We have a table

Student (pno, studno, lastname, firstname, email)

The table is stored as a heapfile. We have a static hash index on **pno**. In addition we have got an unclustered B+-tree on **lastname**. Each record in the hashindex is on the format **(pno, RecordID)** and each leaf-level record in the B+-tree is on the format **(lastname, RecordID, RecordID, ...),** i.e., a variable number of RecordIDs per lastname.

The number of students in the table is 50 000. The heap file contains 1000 blocks. The B+-tree contains 500 blocks on the leaf level and it has 3 levels. The hash index contains 300 blocks and 60 overflow blocks and the average number of blocks accessed per lookup is 1.2.

We want both **studno** and **email** to be unique attributes. How would you ensure that this is efficiently imposed by the database?

Fill in your answer here

Dette kan effektivt overholdes ved å indeksere på de. En indeksering på de vil kreve at ved enhver ny student som skal legges til vil det oppstå feil om den studenten blir gitt et brukt studno eller en allerede eksisterende email.

Attaching sketches to this question?

Use the following code:

1 7 1 2 1 3 7

19

Problem 19 (5 %)

We have the following sequence of operations which arrives at the database. Show how this sequence is executed given that we use two-phase locking of the the type rigorous and using deadlock detection.

r1(X); r2(X); w1(X); r3(Z); r2(Z); w2(Y); c1; c2; w3(Z); c3;

Fill in your answer here

T_1	T_2	T3
rl(x)		
r(x)		
	rl(x)	
	r(x)	
wait for x		
		rl(z)
		r(z)
	rl(z)	
	r(z)	
	wl(y)	
	w(y)	
	commit	
	unlock(x,y,z)	
wl(x)		
w(x)		
commit		
unlock(x)		
		wl(z)
		w(z)
		commit
		unlock(z)

Attaching sketches to this question?
Use the following code:

6 1 3 9 6 8 3

20 Problem 20 (5 %)

We have the following sequence of operations which arrives at the database. Show how this sequence is executed given that we use two-phase locking of the the type rigorous and using deadlock detection.

r1(X); r2(Y); r3(Z); w1(Y); w2(Z); w3(X); c1; c2; c3;

Fill in your answer here

T_1	T_2	T_3
rl(x)		
r(x)		
	rl(y)	
	r(y)	
		rl(z)
		r(z)
wait for y held by T_2		
	wait for z held by T_3	
		wait for x held by T1

Her får vi en deadlock og en av transaksjonene må aborteres.

Attaching sketches to this question?
Use the following code:

1 5 5 1 8 9 8

21 Problem 21 (5 %)

Assume ARIES recovery.
After a crash we found the following log:

LSN	PrevLSN	TransId	Type	PageId
10	0	T1	update	A
11	0	T2	update	B
12	10	T1	commit	
13			begin_checkpoint	
14			end_checkpoint	
15	0	T3	update	B
16	11	T2	update	C

17	16	T2	commit

The following transaction table was found in the log record with LSN 14:

TransId	LastLSN	Status
T1	12	commit
T2	11	active

The following Dirty Page Table (DPT) was found in the log record with LSN 14:

PageId	RecLSN
A	10
B	11

Show the state of the DPT and the transaction table after the analysis has finished.

Fill in your answer here

TT

TransID	LastLSN	Status
T1	12	Commit
T2	17	Commit
T3	15	In progress

DPT

PageID	RecLSN
A	10
B	11
C	16

Attaching sketches to this question?
Use the following code:

2 4 1 8 7 4 1

22 Problem 22 (5 %)

Assume ARIES recovery.

After a crash we found the following log:

LSN	PrevLSN	TransId	Type	PageId
10	0	T1	update	A
11	0	T2	update	B
12	10	T1	commit	
13			begin_checkpoint	

14			end_checkpoint	
15	0	T3	update	B
16	11	T2	update	C
17	16	T2	commit	

The following transaction table was found in the log record with LSN 14:

TransId	LastLSN	Status
T1	12	commit
T2	11	active

The following Dirty Page Table (DPT) was found in the log record with LSN 14:

PageId	RecLSN
A	10
B	11

For each update log record in the log explain whether or not REDO is performed. Also explain why / why not redo is done when the blocks (PageId) have the following PageLSNs:

PageId	PageLSN
A	7
B	15
C	8

Fill in your answer here

Fra forrige spørsmål har vi dette:

TT

TransID	LastLSN	Status
T1	12	Commit
T2	17	Commit
T3	15	In progress

DPT

PageID	RecLSN
A	10
B	11
C	16

Dette gir oss:

LSN	Page affected	ReDo?	Hvorfor/Hvorfor ikke
10	A	YES	A is in DPT, its RecLSN(10) <= LSN(10), and PageId(7) < LSN(10)
11	B	NO	pageid >= LSN
15	B	NO	PageID >= LSN
16	C	YES	C is in DPT, its RecLSN(16) <= LSN(16), and PageId(8) < LSN(16)

Attaching sketches to this question?
Use the following code:

9 8 0 6 9 6 5