Please fill in your name and registration number (Matrikelnr.) **immediately**.

EXAM ON	**MUSTERLÖSUNG**	03.12.2021
◯ DATENMODELLIERUNG (**184.685**)	◯ DATENBANKSYSTEME (**184.686**)	B

Matrikelnr.	Last Name	First Name

Duration: 80 minutes. Provide the solutions at the designated pages; solutions on additional sheets of paper are not graded. **Have a successful exam!**

Question 1: (7)

a) For the relational schemas (R, F_1) and (R, F_2), where $R = ABCDEFG$, find *all keys*.

(4 points)

FDs	Keys
$F_1 = \{FA \to DB, GB \to AE, BA \to FG, CE \to GB\}$	`GBC, FAC, BAC, CE`
$F_2 = \{FG \to D, DE \to GB, A \to FE\}$	`GAC, DAC` ....

b) Consider the relational schemas (R, F_1) and (R, F_2), where $R = ABCDEFG$. Determine their corresponding normal forms, and mark the right answers.

(3 points)

Dependencies	Keys
$F_1 = \{FDE \to BC, FC \to D, FCE \to GA, BAC \to D\}$	`FDE`, `FCE`

neither 3NF nor BCNF ◯   3NF & not BCNF ⊗

BCNF & not 3NF ◯   3NF & BCNF ◯

Dependencies	Keys
$F_2 = \{FGD \to E, GDE \to BA, DAE \to FG, BAE \to GC\}$	`FGD` , `GDE` , `DAE`

neither 3NF nor BCNF ⊗   3NF & not BCNF ◯

BCNF & not 3NF ◯   3NF & BCNF ◯

Attention: for each correct solution: 1.5 point, for each wrong solution: -1.5 point, unanswered questions give 0 points. In total you get at least 0 points.

Question 2: (10)

Assume that a college organizes its data in the following schema (primary keys are underlined):

Program(<u>ProgID</u>, Name, Type)
Course(<u>CourseID</u>, CourseName, Semester, *ProgID: Program.ProgID*)
attended(<u>*StudentID*</u> *Student:StudentID*, <u>*CourseID*</u>*: Course.CourseID*,Grade)
Student(<u>StudentID</u>, Lastname, Givenname)

a) Consider the folloing query given in **relational algebra**.
Describe *briefly* (**1 short sentence!**) what values are returned by the query. (3 Points)

$$\pi_{\text{Lastname}}(\texttt{Student} \bowtie \pi_{\text{StudentID}}(\texttt{attended} \div \pi_{\text{CourseID}}(\texttt{course} \bowtie \sigma_{\text{Name}=`ComputerScience' \wedge \text{Type}=`Bachelor'}(\texttt{Program}))))$$

```
Lastname of the students that have visited all courses in the program Bachelor in
Computer Science.
```

Name of the Students $\boxed{+1}$
Alle courses $\boxed{+1}$
Computer Science $\boxed{+0.5}$
Bachelor $\boxed{+0.5}$

b) Consider the following query in the **domain calculus**.
Determine the result of this query on the database instance given below. (3 Points)

$$\left\{ \left[\text{studentid}, \text{lastname}, \text{grade}\right] \;\middle|\; \left[\text{studentid}, \text{lastname}\right] \in \texttt{Student} \wedge \exists \text{cid} \left[\text{studentid}, \text{cid}, \text{grade}\right] \in \texttt{attended} \wedge \text{grade} = \text{'A+'} \right\}$$

Announced Correction: [**studentid,lastname,givenname**] $\in$ **Student Does not affect the solution.**
Database Instance:

Student		
StudentID	**Lastname**	**Givenname**
1	Aiken	Howard
2	Babbage	Charles
3	Cook	Stephen
4	Dijkstra	Edsger
5	Engelbart	Douglas

attended		
StudentID	**CourseID**	**Grade**
1	Electrical Engineering	A+
3	Cryptography	A-
1	Math 101	B
3	Graph Theory	A+
2	Algebra	A+
2	Complexity Theory	A-

studentid	lastname	grade
1	Aiken	A+
2	Babbage	A+
3	Cook	A+

Select on grade 'A+' $\boxed{+1}$
Implicit join on studentid $\boxed{+1}$
Attributes $\boxed{+0.5}$
All values $\boxed{+0.5}$

c) Formulate a query in the **tuple calculus** that does the following:
Output last name and first name of students, which have at least one unfinished course (not attended) in their *Bachelor* studies in program *Computer Science*. (4 Points)

$$\Big\{ [s.lastname, s.givenname] \;\Big|\; \exists s \in \texttt{Student}\Big(\exists c \in \texttt{Course}\Big(\exists p \in \texttt{Program}\big(c.\texttt{ProgID} = p.\texttt{ProgID} \wedge p.\texttt{Type} = \texttt{'Bachelor'} \wedge p.\texttt{Name} = \texttt{'Computer Science'} \wedge \neg\exists a \in \texttt{attended}\big(a.\texttt{CourseID} = c.\texttt{CourseID} \wedge s.\texttt{StudentID} = a.\texttt{StudentID}\big)\big)\Big)\Big)\Big\}$$

Relevant relations exist $\boxed{+1}$
Where statement auf Computer Science and Bachelor $\boxed{+1}$
Negation on attended exists $\boxed{+1}$
Joins on StudentID, CourseID, ProgID (0.5, if only 2 stated) $\boxed{+1}$
Attributes in output tuples missing $\boxed{-0.5}$
Exists missing $\boxed{-0.5}$
Relational Algebra / Tuple / Domain calculus swapped; but otherwise stated correct $\boxed{-1.5}$

Question 3: (10)

a) Consider the relational schema (R_d, F_d), where $R_d = ABCDEFG$, together with all keys are given below.

Using the (Relational) Synthesis Algorithm ("Synthesealgorithmus"), find a lossless and dependency preserving decomposition in 3NF (F_d is already in a canonical form/is a minimal cover).

For each schema R_i of the decomposition, state its attributes and mark exactly one key by underlining.

(6 points)

$$F_d = \{F \to D, FG \to E, FB \to G, GD \to B, E \to FGA\}, \text{Schlüssel} = \{FGC, FBC, CE\}$$

Decomposition into 3NF (Underline one key in each relation)

R1 $\underline{\texttt{FGC}}$	R2 $\underline{\texttt{FG}}\texttt{B}$	R3 $\underline{\texttt{FG}}\texttt{AE}$	R4 $\underline{\texttt{F}}\texttt{D}$	R5 $\underline{\texttt{GD}}\texttt{B}$ ....

b) Consider the relational schema $R = (R_d, F_d)$ where $R_d = ABCDEF$ and $F_d = \{BDE \to AF, CEF \to AD\}$, with keys $\{BCDE, BCEF\}$.

You are given the following subschemas R_i of R:

relational schema	non-trivial functional dependencies	keys
$R_1 = ACDEF$	$C_1 = \{CEF \to AD\}$	CEF
$R_2 = ABDEF$	$C_2 = \{BDE \to AF\}$	BDE
$R_3 = BCDE$	$C_3 = \emptyset$	$BCDE$

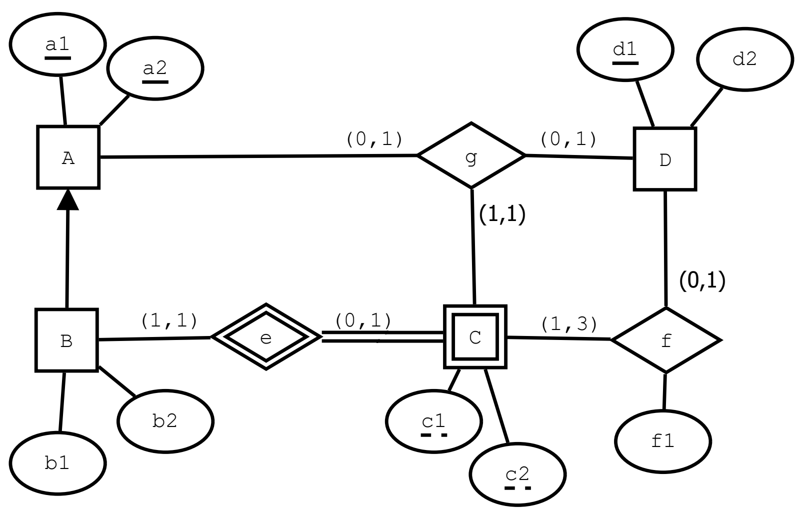
Determine for the following decompositions of R_d, F_d whether the decomposition is dependency preserving. If the decomposition is not dependency preserving, state at least one (non-trivial) functional dependency that was lost. In addition to this, you also need to answer whether the decomposition is lossless or not.

(4 points)

decomposition	dependency preserving		"lost" FDs	lossless	
(R_1, R_2)	⊗ yes	○ no	....	○ yes	⊗ no
(R_2, R_3)	○ yes	⊗ no	$CEF \to AD$	⊗ yes	○ no

Question 4: (6)

Construct a relational schema according to the EER-diagram given below. For each relation, clearly mark the primary key by underlining the corresponding attributes. Mark foreign keys (FK) either by prefixing the name of the relation referenced by the FK (i.e., by `Relation.Attribute`) or by using the notation `NameOfAttribute:Relation.Attribute` (where `NameOfAttribute` is the name of the attribute in the current schema and `Relation.Attribute` describes the value that is referenced by the FK). You do not need to distinguish between FKs consisting of a single attribute and FKs combining several attributes.
Create as few relations as possible without introducing any redundancies. Note that the database does not allow `NULL`-values.



A ($\underline{\text{a1, a2}}$ ....)

D ($\underline{\text{d1}}$, d2 ....)

B ($\underline{\textit{a1:A.a1, a2:A.a2}}$, b1, b2 ....)

C ($\underline{\textit{a1:B.a1, a2:B.a2,}\text{ c1, c2}}$, gd1:D.d1, ga1:A.a1, ga2:A.a2 ....)

f (*a1:C.a1, a2:C.a2, c1:C.c2, c2:C.c2,* $\underline{\textit{d1:D.d1}}$, f1 ....)

(....)

(....)

Question 5: (8)

Consider the relational schemas $X(\underline{A}BC)$, $Y(\underline{D}E)$, and $Z(\underline{AC}E)$. Assume there exists an instance of X containing 2 tuples, an instance of Y containing 3 tuples, and an instance of Z containing 4 tuples. Thus

$$X(\underline{A}BC)\colon 2 \qquad\qquad Y(\underline{D}E)\colon 3 \qquad\qquad Z(\underline{AC}E)\colon 4$$

Consider the expressions in Relational Algebra given below. For these expressions, provide the minimal and maximal possible size (= number of tuples) of their results over instances for X, Y, and Z of the given sizes. In addition, provide concrete instances over which the expressions actually realize these bounds, i.e. return results of minimal/maximal size. Make sure that the provided instances contain exactly the given number of tuples.

Attention: Points for correct instances are awarded only if the stated corresponding size is also correct!

a) **Expression:** $\pi_{A,D}(Y \bowtie \rho_{A\leftarrow D} Y) - \pi_{A,D}(X \bowtie_{X.A=Y.D} Y)$

min. size of the result: 1 ....

X

A	B	C
1	–	–
2	–	–

Y

D	E
1	1
2	2
3	3

max. size of the result: 9 ....

X

A	B	C
1	–	–
2	–	–

Y

D	E
3	6
4	6
5	6

b) **Expression:** $\pi_{A,D,E}(Y ⟖ Z) \cup (Y \bowtie \rho_{A\leftarrow E} Y)$

min. size of the result: 3 ....

Y

D	E
1	4
2	5
3	6

Z

A	C	E
4	1	4
5	2	5
6	3	6
7	4	7

max. size of the result: 15 ....

Y

D	E
1	4
2	4
3	4

Z

A	C	E
5	6	4
7	8	4
9	10	4
11	12	4

Question 6: (9)

To avoid mistakes and wrong assignments, the orginizers of some workshops have decided to use a database instead of handwritten notes. You were chosen to develop the database. The database is supposed to store information about the workshops, participants and speakers.

Create an EER-diagram based on the information described below. Use the (min,max) notation, and in case no explicit information is given, assume that there are no restrictions on the values for (min,max). The model shall work without using `NULL`-values, redundancies shall be avoided, and it is not allowed to introduce any attributes not described by the text. Finally, make sure that a key is defined for each entity type.

For persons, a distinction is made between speakers and workshop participants. For each person an identification (id), the name (name), and the affiliation to a university (affiliation) have to be stored, where the identification of the person is unique.

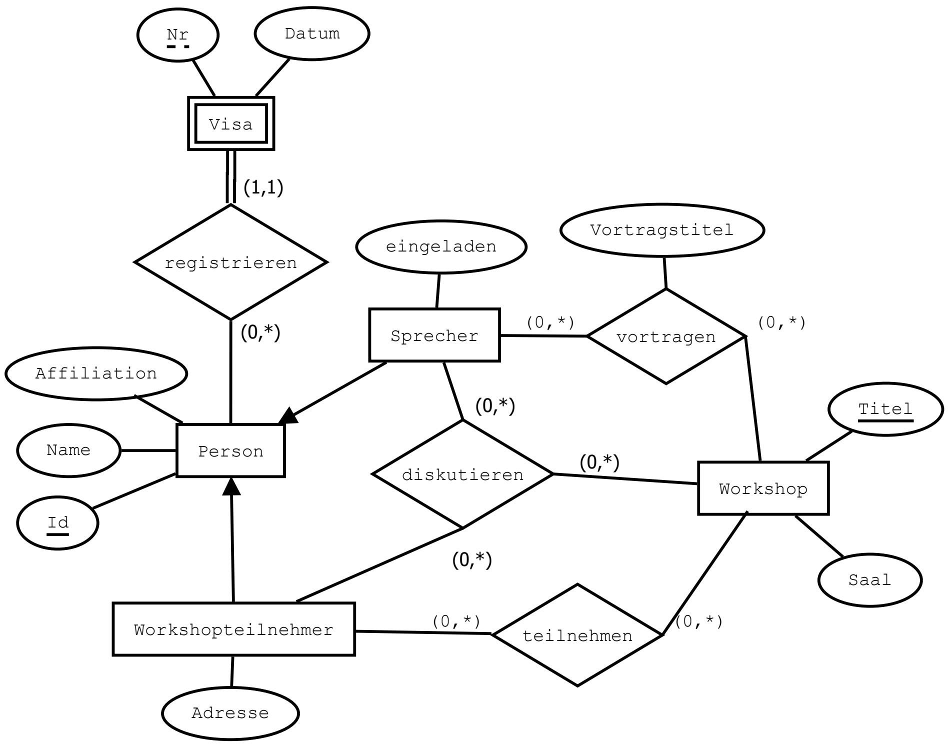
In order to be able to reach the workshop participants by post if necessary, the addresses (address) should also be stored. Any number of workshop participants can take part in any number of workshops.

A workshop always has a unique title (title) and also a hall (hall) in which the workshop takes place has to be stored.

In contrast to workshop participants, we do not need the addresses for speakers, but for speakers we have to be able to set a flag, which indicates whether they have been invited to a workshop (invited). Any number of speakers can lecture at any number of workshops, and for each lecture the title of the lecture has to be stored.

Moreover, workshop participants can discuss with speakers at workshops.

It is important that some speakers or workshop participants have to register for a visa in order to be able to participate in a workshop at all. It has to be clear which speaker or participant has registered for a Visa. A visa can be uniquely identified by the person who carried out the registration and an internal number (nr). In addition, the date (date) of the application should be stored for each visa.



Nr
Datum
Visa
(1,1)
registrieren
(0,*)
eingeladen
Vortragstitel
Sprecher
(0,*)
vortragen
(0,*)
Affiliation
Person
Name
Id
(0,*)
diskutieren
(0,*)
Workshop
Titel
Saal
(0,*)
Workshopteilnehmer
(0,*)
teilnehmen
(0,*)
Adresse

Question 7: (10)

Given the following relational schema:

```
Patient(pid, lastname)
Diagnose(caseid, pid: Patient.pid, ICD10)
Hospitalization(caseid: Diagnose.caseid, admitted, discharged)
```

In addition, given the following database instance:

`patient`:

pid	lastname
1	Meier
2	Hacker
3	Jensen
4	Lang
5	Mayer
6	Gruber
7	Mayr
8	Nilsson
9	Weiss
10	Fuchs
11	Yıldırım
12	Abramowitz

`diagnose`:

caseid	pid	ICD10
1	1	A08.5
2	5	O80
3	1	A08.1
4	4	I10.0
5	6	S20.7
6	9	S22.41
7	10	S41.1
8	8	S81.0
9	2	S81.87
10	3	S82.7
11	7	T33.5
12	12	S86.7
13	11	S83.53
14	1	S83.14

`hospitalization`:

caseid	admitted	discharged
14	2021-01-03	2021-01-06
5	2021-01-07	2021-01-09
6	2021-01-04	2021-01-11
9	2021-01-07	2021-01-10
10	2021-01-11	2021-01-18

a) Evaluate the following SQL Query. (4 Points)

```sql
SELECT patient.lastname,
       diagnose.ICD10,
        EXTRACT(DAY FROM hospitalization.discharged- hospitalization.admitted) AS days
FROM hospitalization
         INNER JOIN diagnose ON hospitalization.caseid = diagnose.caseid
         INNER JOIN patient  ON diagnose.pid = patient.pid
WHERE hospitalization.admitted BETWEEN '2021-01-02' AND '2021-01-05'
ORDER BY days;
```

Remark:
`EXTRACT(DAY FROM hospitalization.discharged - hospitalization.admitted) AS days` returns the difference between two dates. For example, on `'2021/01/03'`, `'2021/01/06'` returns 3.

Result of the query (Hint the result has 2 rows):

lastname	icd10	days
Meier	S83.14	3
Weiss	S22.41	7

R#1 (lastname): Meier	+0.5
R#2 (lastname): Weiss	+0.5
R#1 (icd10): S83.14	+0.5
R#2 (icd10): 22.41	+0.5
R#1 (days): 3	+0.5
R#2 (days): 7	+0.5
Systematic off by 1 days	-0.5
Values swapped	-0.5
Wrong order	-0.5
$\sum$	+4

b) Is it possible that there are two diagnoses (diagnose.ICD10) for one patient at one admission date (hospitalization.admitted)? (1 Point)

⊗ yes ◯ no

c) Briefly explain whether you think that the property at Question 7(b) is appropriate? What might be eventual disadvantages? (Restrict your answer to at most 2 sentences.) (1 Point)

1 point for thinking about the modelling, e.g.,

Yes, a patient has one primary reason to be treated	+1
No, a patient can have many reasons for treatment	+1
Disadvantage, Duplicates	+1
Advantage, multiple diagnoses for each patient	+1

d) Provide an SQL query for the following task:
List all patients (lastname) with Diagnose (icd10) which show injuries of the knee or lower leg (icd10 is between S80 and S89). Sort by patient name. (4 Points)

```
select lastname
from patient inner join diagnose d on patient.pid = d.pid
where icd10 like 'S8%';
```

Where statement exists Like	+1
Where statement exists with reasonably sufficient alternative (OR/between)	+1
Join patient and diagnose, tables	+0.5
Join (ON oder =)	+0.5
Select contains lastname	+0.5
Select contains icd10	+0.5
Sorting	+0.5
Too many attributes or tables	-0.5
$\sum$	+4

Have a successful exam!

Overall: 60 points