Exercise 05

Relational algebra and functional dependencies

Task 1: Relational algebra

Given the following relations:

Studenten:

SName	SMatrikel	SGeburtstag
Heintje	2143	1900-01-01
Eva	3333	1900-01-01
Luise	3334	1990-03-01
Daniel	3335	1990-04-02
Daniel	3336	1990-10-10
Heintje	3337	1990-10-10

Dozenten:

DName	DBuero	DTel
Klaus	C201	123
Maria	D22	NULL
Marlene	C204	443
Matze	E4	NULL

Veranstaltungen:

VName	VSemester	VRaum	VDozent
Beachvolleyball	ss17	Strand	Maria
Beachvolleyball	ss18	Strand	Maria
Drachenfliegen	ss17	Strand	Maria
Drachenfliegen	ss18	Strand	Maria
Sackhüpfen	ws17	NULL	Klaus
Sackhüpfen	ws18	NULL	Klaus
Tanzgymnastik	ss18	D111	Klaus
Tanzgymnastik	ws17	D111	Klaus

a) Specify the formal relational schemas and the mapping *dom : U → D* for the relations Students and Lecturers. To do so, you can define your own value ranges or reference suitable ones from TSQL.
Students={SName,SMatriculation,SDate_of_birth}
Lecturers ={LName,LOffice,LTel} dom(SName)=dom(LName)=dom(LOffice)=dom(LTel)=varchar(30),
dom (SMatriculation)= decimal(4,0),
dom (SDate_of_birth)=date

b) Specify the formal database schema *S*.
S={Students,Lecturers,Events}

c) What is *r(Lecturers)*?
r(Lecturers) = (t_1, t_2, t_3, t_4)

d) What is t_2*(ESemester)* of *Events*?
t_2(ESemester)=ss18

e) Specify a deletion anomaly, insertion anomaly and update anomaly for the relation Lecturers.
Even such small examples can already lead to problems with redundancies. Saving Offices in the same relation as Lecturers and not moving Offices into a separate relation results in a deletion anomaly, for example when the Lecturer Maria is deleted. The information that the Office D22 even exists is lost in this case. In addition, the relation Lecturers is not suitable for storing new Offices without Lecturers. In that case, the Lecturer would always be `null` and we want the Lecturers to be the primary key. Thus, despite the `Office` attribute, we cannot add new (empty) Offices (insertion anomaly). If we remodel our buildings and the Offices identifiers change, for example now C201 is suddenly called C2.2.1, then we generate an update anomaly if we only change the identifier for Klaus' office, but not for Marlene's.

Task 2: Keys and superkeys

Justify the following statements and show them formally by applying the definitions:

a) `SName` in `Students` cannot be a key.
Daniel appears twice, but has a different matriculation number. Formally, a key is a minimal identifier attribute set. This means that every subset of
$X \subseteq$ {SName} $\subseteq$ Students must differ in at least one attribute value. But t_4(SName)=t_5(SName)=Daniel.

b) `LName` from `Lecturers` does not meet the foreign key constraint for `ELecturer` in `Events`.
It's exactly the wrong way round. ELecturer points to LName. Formally: {t(LName) | t $\in$ Lecturers} $\subseteq$ {t(ELecturer) | t $\in$ Events} must apply. However, this is not fulfilled for t_4(LName)=Matze.

c) `ELecturer` in `Events` fulfils the foreign key constraint for `LName` from `Lecturers`.
This way is correct because {t(ELecturer) | t $\in$ Events} $\subseteq$ {t(LName) | t $\in$ Lecturers}

d) `ESemester`, `ERoom`, `ELecturer` in `Events` is not an identifier attribute set of `Events`.
Maria has hang-gliding and beach volleyball on the beach in the summer semester 2018. Formally:
{ESemester, ERoom, ELecturer} $\subseteq$ Events always differ in at least one attribute value. But t_2(ESemester, ERoom, ELecturer) = t_4(ESemester, ERoom, ELecturer)

e) `LName`, `LOffice` is not a minimal identifier attribute set of `Lecturers`.
{LName, LOffice} is an identifier attribute set because { LName, LOffice } $\subseteq$ Lecturers always differ in at least one attribute value. {LName, LOffice} is minimal if there is no real subset $X \subset$ {LName, LOffice}, so that X is once again an identifier attribute set. However, LName is such a subset, so {LName, LOffice} is not a minimal identifier attribute set.

f) `LOffice` in `Lecturers` fulfils the key constraint.
See previous solution, LOffice is a single-element identifier attribute set, thus minimal and it therefore fulfils the key constraint. We cannot tell from the data that LOffice is a key, that is domain knowledge.

g) The relational schema `R` is a superkey of `R`.
Every relational schema R=A_1, .., A_n is an identifier attribute set of R, since A_1, .., A_n always differs in at least one attribute value (set semantics of relational algebra)

h) `EName`, `ESemester` is a superkey of `Events`.
{EName, ESemester} is an identifier attribute set of Events because there are no two tuples with
t_i(EName,ESemester) = t_j(EName,ESemester) in Events.

Task 3: Functional dependencies

Starting from our example using Lecturers, Students, Events and Grades which students can get in the events, we now assume that there are only the following attributes: `SName`, `SMatriculation`, `SDate_of_birth`, `LName`, `LOffice`, `LTel`, `EName`, `ESemester`, `ERoom` and `Grade`. We can already store the required data with these attributes, for example in a single relation `RData`:

SName	SMatrikel	SGeburtstag	DName	DBuero	DTel	VName	VSemester	VRaum	Note
Eva	3333	1900-01-01	Maria	D22	NULL	Beachvolleyball	ss18	Strand	4.0
Eva	3333	1900-01-01	Maria	D22	NULL	Drachenfliegen	ss17	Strand	NULL
Luise	3334	1990-03-01	Maria	D22	NULL	Beachvolleyball	ss17	Strand	4.0
Luise	3334	1990-03-01	Maria	D22	NULL	Beachvolleyball	ss18	Strand	2.0
Luise	3334	1990-03-01	Maria	D22	NULL	Drachenfliegen	ss18	Strand	NULL

a) What functional dependencies between these attributes do you know from the domain knowledge of the previous exercises?
Matriculation → SName, SDate_of_birth
LName → LOffice, LTel
EName, ESemester → ERoom
Matriculation, EName, ESemester → Grade

b) Simplify this set F using the SPLITTING algorithm.
On the right, everything that is single-element and trivial will disappear:
Matriculation → SName,
Matriculation → SDate_of_birth
LName → LTel
LName → LOffice
EName, ESemester → ERoom
Matriculation, EName, ESemester → Grade

c) Determine *{F}⁺* for the functional dependencies *F* identified in part (a) of the task.
Difficult, because now we must try out all the combinations.
Trivial single-element: Matriculation → Matriculation, SName → SName, SDate_of_birth → SDate_of_birth, ..
Trivial two-element: Matriculation, SName → Matriculation, SName, ..
Single-element: Matriculation → SName, Matriculation → SDate_of_birth, LName → LTel, ..
Two-element: Matriculation, SName → SName, Matriculation, SDate_of_birth, SDate_of_birth, ..,
EName,ESemester → ERoom, ..
Three-element: Matriculation, SName, SDate_of_birth → SName, .., Matriculation, EName, ESemester → Grade
Four-element: …
…

d) What is $\{SMatriculation\}^+_F$?
$\{SMatriculation\}^+_F$ = {SMatriculation, SName, SDate_of_birth}

e) What is $\{SMatriculation, LName\}^+_F$?
$\{SMatriculation,LName\}^+_F$ ={SMatriculation, LName, SName, SDate_of_birth, LOffice, LTel}

f) What is $\{SName, LOffice, ERoom\}^+_F$?
$\{SName, LOffice, ERoom\}^+_F$ ={SName, LOffice, ERoom}

g) Simplify the set from part (b) of the task further using the COVER algorithm.
SPLITTING see part (b) of the task. F_C=SPLITTING(F)

Minimise left sides, we do not need to check single-element sets
EName, ESemester → ERoom
EName $\in \{ESemester\}^+_{FC}$? No.
ESemester $\in \{EName\}^+_{FC}$? No.

Matriculation, EName, ESemester → Grade
Matriculation $\in$ {EName, ESemester}$^+_{FC}$? No.
EName $\in$ {Matriculation, ESemester}$^+_{FC}$? No.
ESemester $\in$ {EName, Matriculation}$^+_{FC}$? No.

Remove unnecessary functional dependencies
Is SName $\in$ {Matriculation}$^+_{\{FC - Matriculation \to SName\}}$? No.
Is SDate_of_birth $\in$ {Matriculation}$^+_{\{FC - Matriculation \to SDate\_of\_birth\}}$? No.
Is LTel $\in$ {LName}$^+_{\{FC - LName \to LTel\}}$? No.
Is LOffice $\in$ {LName}$^+_{\{FC - LName \to LOffice\}}$? No.
Is ERoom $\in$ {EName, ESemester}$^+_{\{FC - EName, ESemester \to ERoom\}}$? No.
Is Grade $\in$ {Matriculation, EName, ESemester}$^+_{\{FC - Matriculation, EName, ESemester \to Grade\}}$? No.

In summary, we can say about this step that no change is to be expected here either, since every element that occurs on a right side of an FD occurs only once on a right side.

Undo splitting, then we are back to part (a) of the task. If you have determined a different set in (a), your algorithm might look different, but the result should be identical.

h) Determine the keys using heuristics.
SPLITTING(F) see (a)

Matriculation → SName, SDate_of_birth
LName → LOffice, LTel
EName, ESemester → ERoom
Matriculation, EName, ESemester → Grade

Attributes that do not appear in any FD: none
Attributes that do not appear on any right side are in the key: Matriculation, LName, EName, ESemester
Test whether keys are found: {Matriculation, LName, EName, ESemester}^+_F={Matriculation, LName, EName, ESemester, SName, SDate_of_birth, LOffice, LTel, ERoom, Grade}=RData
Key found, {Matriculation, LName, EName, ESemester} is the only key of RData