Exercise sheet 1 (WS 2019) – Sample solution

3.0 VU Data Modelling / 6.0 VU Database Systems

About the exercises

General information

In this part of the exercises you are asked to create a small database using EER-diagrams, transform EER-diagrams into a relational schema and make yourself familiar with relational algebra and relational calculus.

We recommend you to solve the problems **on your own** (it is a good preparation for the exam – and also for your possible future job – to carry out the tasks autonomously). Please note that if we detect duplicate submissions or any plagiarism, both the "original" and the "copy" will be awarded 0 points.

Your submission must consist of a single, typset PDF document (max. 5MB). We do not accept PDF files with handwritten solutions.

In total there are 8 tasks and at most 15 points that can be acchieved on the entire sheet.

Deadlines

until 30.10.	**12:00pm**	Upload your solutions to TUWEL
13.11.	13:00pm	Evaluation and feedback is provided in TUWEL

Consultation hours (optional)

In the week before the deadline there are consultation hours held by tutors. If you have problems understanding the topics of the exercise sheet or questions about the exercises, you are welcome to just drop in at these consultation hours. The tutors will gladly answer your questions and help you understand the subjects.

The goal of these consultation hours is to help you with **understanding the topics and specific tasks** of the exercise sheet. The tutors will not solve your exercises or check your answers before you hand them in.

Participation is completely voluntary — dates and locations of the consultation hours can be found in TUWEL.

Debriefing (optional)

A few days after you received your feedback and grading of this exercise sheet, there is the opportunity to go through the tasks in small groups (max. 25 persons). The (relative) small group size is intended to enable an active participation. Each of these groups will be held by an assistant. The specific agenda of these meetings will depend on the interests and question of the participants (i.e., you). The main objectives are answering your open questions and resolving your remaining issues regarding the exercises. Therefore, please have a look at your feedback and evaluation to identify such problems before you attend the class. When participating, dare to ask your questions – no question has a (negative) impact on your grade.

Participation is absolutely voluntary. Registration in TUWEL is required to keep the size of the groups small. Dates and locations can be found in TUWEL.

Further questions – TUWEL forum

If you have any further questions concerning the contents or organization, do not hestitate to ask them on TUWEL forum.

Exercise: EER-diagrams

Exercise 1 (Creating an EER-diagram) [3 points]

NASA and ESA are planning a joint acquisition of their most important hardware components. In order to make the best possible decision in this sensitive matter, a database containing the most important information about the possible opportunities is needed. After an intensive worldwide selection process, you are hired to create the database and you receive the result of the requirements analysis as given on the next page.

Draw an EER-diagram based on the available information (see next page). Use the notation presented in the lecture and the (min, max) notation. NULL values are not allowed and redundancies should be avoided. Sometimes it might be necessary to introduce additional keys.

A possible software for creating the EER-diagram is DIA (`http://wiki.gnome.org/Apps/Dia`, binaries at `http://dia-installer.de`; Attention: select ER in the diagram editor!). Of course you are also allowed to create the diagram with any other suitable software.

Description of the issues:

Both coffee machines and coffee capsules are products that have a price (PRICE) and a model number (MODELNR), where the model number is unique for each product. For coffee machines, the name (NAME), maximum water pressure (PRESSURE), and expected durability, measured in number of prepared portions (NUMBCAPS), are also collected for decision making. Additionally, it should also be noted if a coffee machine is an advancement of (at most one) older machine. For coffee capsules, the material (MATERIAL) of the capsule is stored, as well as the unique type of coffee capsule. Capsule types have a volume (VOLUME) and can be uniquely identified by the combination of size their (SIZE) and ID (CTID). A capsule type may be compatible with other capsule types. Each coffee machine supports at least one capsule type, but not more than ten.

Products are produced by manufacturers. For each manufacturer, the country in which he is taxable (TAXC) is noted, as well as the unique brand name (BRAND). Moreover, an arbitrary number of reviews are collected for each manufacturer. The reviews for each manufacturer are numbered consecutively (ID) and the rating is saved (TXT). Some coffee machines require licenses to support certain capsule types. In such cases, for each affected coffee machine it should be saved for which capsule types licenses of which manufacturers were purchased and the respective licence fee (FEE).

Products are sold by retailers. For each retailer, the name (NAME) and the website (WEBSITE) are known. Both the name and the website of all the retailers are different. Retailers often offer bulk discounts. The different bulk discounts of a retailer can be distinguished by the combination of label (LABEL) and allowed discount (DISCOUNT). There is, however, no coordination among the retailers in assigning labels and discounts. In order to benefit from a discount, a minimum quantity (QUANTITY) of the selected product has to be purchased. Discounts are only granted after buying at least four different products, and the minimum selling quantity of each product may vary. In addition, there may be coupon codes for a bulk discount. Such codes (CODE) are unique in combination with the discount.

For each coffee the aroma (AROMA), the acid (ACIDITY) as well as the body (BODY) is noted. The different coffees should be distinguished by their name (NAME). It is also stored if the coffee is a ristretto, lungo or espresso. Additionally, for each ristretto the roasting temperature (ROASTTEMP) and for each espresso the brew pressure (PRESSURE) is saved.

Finally, for each coffee capsule, the contained coffee is noted, as well as for each coffee machine which coffee they can prepare (each coffee machine can prepare at least one coffee).

Solution: See Figure 1.

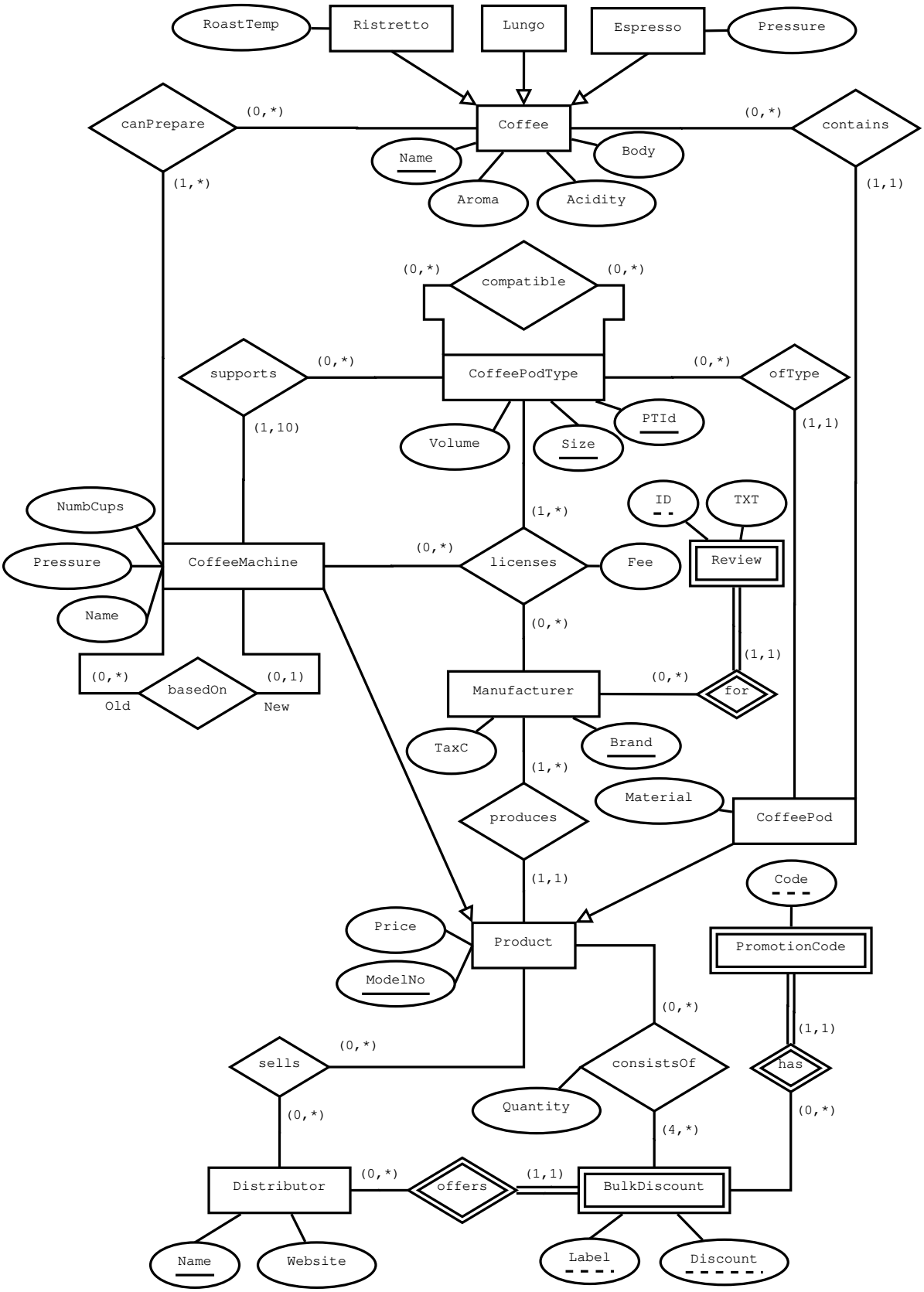


Figure 1: Solution of Exercise 1

Exercise 2 (Semantics of EER diagrams) [1 point]

Consider the EER-diagram shown in Figure 2. It illustrates an arbitrarily chosen variant of the so called *property graph* model. (Background: property graphs are a further popular data model, especially suitable for storing and processing network data. However, to solve this problem, there is no knowledge about property graphs necessary. This task is only about the information presented in the EER diagram.)

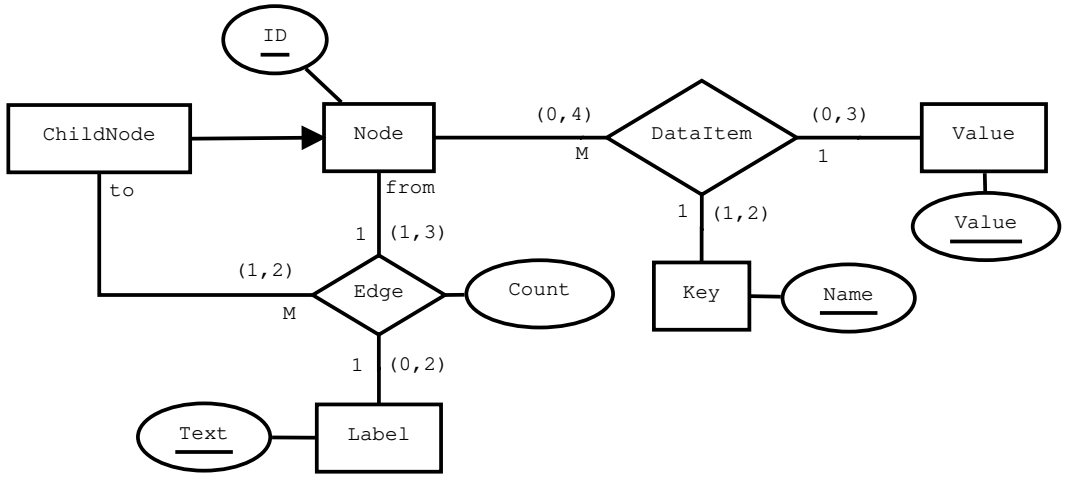


Figure 2: EER-diagram for exercise 2

1. In the ER diagram, both the notation by means of functionalities, as well as the (min, max) notation is used.
   (*note:* this is not commonly done in practice.)

   Therefore, the diagram contains more information compared to the use of only one notation.

   - Specify a specific relationship in the diagram where omitting one of the two notations causes a loss of information.
   - For the chosen relationship type, omitting which notation leads to the loss of information?
   - Explain briefly in your own words which information can no longer be represented.
   - Provide a concrete example of the lost information, i.e. for the type of relationship you have chosen, specify an instance that violates (at least) one condition expressed by the omitted notation, but satisfies all requirements by the remaining notation.

   Solution:
   Yes, both notations carry information that cannot be expressed by the other notation. In the following, we only provide one such example for each notation, but there are more such information in the diagram:

   - If for the relationship type `Edge` the information (1:1:M) is dropped, then the (min,max)-notation can neither express that each edge (= each pair of a `from` and a `to` node) must have a distinct label, nor that no `ChildNode` can have two

or more incoming edges with the same label from different nodes (each combination of `ChildNode` and `Label` must be related to at most one `Node` via an `Edge`-relationship).

A concrete counter example would be nodes v_1 und v_2, with v_1 also being a `ChildNode`, and a `Label` "Edge". By the (min,max)-notation, the following would be a valid instance of the relationship-type `Edge`: The instance contains two directed edges $e_1 = (v_1, v_1)$ and $e_2 = (v_2, v_1)$. Both edges get assigned the `Label` "Edge" and the value of `count` is 1. This is forbidden byt the (1:1:M), since now two entities of type `Node` are in an `Edge`-relationship with the pair (v_1,"Edge") of entity v_1 and "Edge". These being the entities v_1 and v_2.

- If the *(min,max)-notation* is dropped for the relationship type `Label`, the remaining cardinality information do neither express that each entity of type `ChildNode` must take part in at least one and at most two relationships of type `Label`, nor that each entity of type `Node` must occur in at least one and at most three of such relationships, and furthermore also the information that each entity of type is allowed to take part in at most two of such relationships is lost.

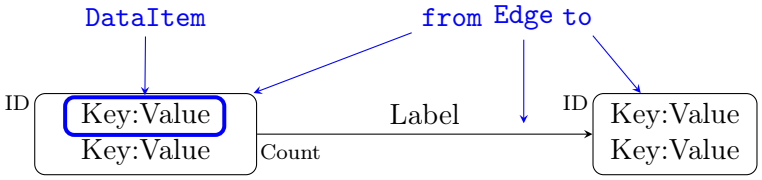
  A concrete counter example would thus be two entities v_1, v_2 of type `Node`, with v_1 being also of type `ChildNode`, as well as an entity "Text" of type `Label`. A set of `Edge`-relationships that contains a single directed edge with the `''Label''` "Text" and a value of 1 for `count` from v_2 to v_1 (i.e., the relationship consisting of a single tuple $(v_2, v_1, \text{"Text"}, 1)$) would satisfy all requirements expressed by the remaining cardinalities, but not the additional requirement expressed by the (min,max)-notation that each node must have at least one outgoing edge (must take the role `from` in at least one `Edge` relationship). The node v_1 violates this condition.

- For the relationship type `DataItem`, an analogous case can be made.

2. Suppose you get the graph shown in Figure 3, and now you want/have to verify if it represents a valid instance of a property graph as described by the EER diagram.

   Describe at least seven (7) violations of the rules presented in the EER diagram. (*Hint:* Specifiy only violations described by facts in the EER diagram. Ignore things that "make no sense" but not declared in the EER diagram.)

   The "mapping" between the entity and relationship types described in the EER diagram and the specific instances in figure 3 is shown in the following graphic:



   Note the following points:

   - Nodes without incoming edges are always entities of type `Node`, but not of type `ChildNode`.
   - Nodes with incoming edges are always of type `ChildNode`.

- Entities of type `Label`, `Key`, and `Value` are identified by the value of their key attributes `Text`, `Name`, and `Value`.
- Semicolons "; " separate different entities of the same type represented by their key.
- Elements shown in the above graphic blue are additional explanations and not part of the graph.

Solution:
The instance contains (at least) the following 16 violations:

- At `Node` $n9$, two entities of type `Value` take part in a `DataItem` relationship. These entities are "secret" and "important". However, the relationship type `DataItem` is ternary, i.e. at each relationship consists of one entity of type `Node`, one of type `Key` and one of type `Value`.
- Besides the two `Node`-entities $n10$ and $n6$, also two `Label`-entities ("next" and "prev") take part on the `Edge` relationship. However, the relationship type `Edge` is ternary and requires the participation of *exactly one* entity of type `Label`.
- The `Label` "next" appears in three `Edge` relationships, these being the edges $n6 \to n9$, $n9 \to n10$, as well as $n10 \to n6$. However, each `Label` is allowed to appear in at most two such relationships.
- The triple $(n2, n6, \text{"docked"})$ occurs twice as an instance of the relationship type `Edge` (both times with different values for `count`). However, each relationship can occur at most once.
- The node $n6$ takes the role `to` in 4 relationships of type `Edge` (i.e., it has four incoming edges). The allowed maximum is 2.
- The node $n11$ takes the role `from` in no relationship of type `Edge` (has no outgoing edge). The allowed minimum is 1.
- The node $n12$ takes the role `from` in no relationship of type `Edge` (has no outgoing edge). The allowed minimum is 1.
- The two relationships $(n4, n8, \text{"selects"})$ and $(n4, n8, \text{"trained"})$ (using the notation $(\texttt{from}, \texttt{to}, \texttt{label})$ violate the function $(\texttt{from}, \texttt{to}) \to \texttt{Label}$ required by the diagram.
- No entity of type `Label` is part of the relationship of type `Edge` between $n7$ and $n10$. Since `Label` is a ternary relationship type that requires the participation of an entity of type `Label`, this is not allowed.
- The two edges (= entities of type `Edge`) $(n7, n12, \text{"achieve"})$ and $(n8, n12, \text{"achieve"})$ violate the required function $(\texttt{to}, \texttt{Label}) \to \texttt{from}$, since they are two edges with different start nodes but the same end node and the same label.
- The `Key` "position" appears in 3 relationships of type `DataItem`, despite the allowed maximum being 2.
- The `Key` "type" appears in 3 relationships of type `DataItem`, despite the allowed maximum being 2.
- The `Key` "is" appears in 3 relationships of type `DataItem`, despite the allowed maximum being 2.
- Der `Node` $n11$ kommt in 5 Beziehungen des Typs `DataItem` vor. Erlaubt sind maximal 4.

- The `Node` $n11$ appears in 5 relationships of type `DataItem`, despite the allowed maximum being 4.
- The two relationships ($n11$, "first", "boss") and ($n11$, "first", "mum") violate the function (`Node`, `Key`) $\rightarrow$ `Value`.
- The two relationships ($n11$, "second", "mum") and ($n11$, "first", "mum") violate the function (`Node`, `Value`) $\rightarrow$ `Key`.

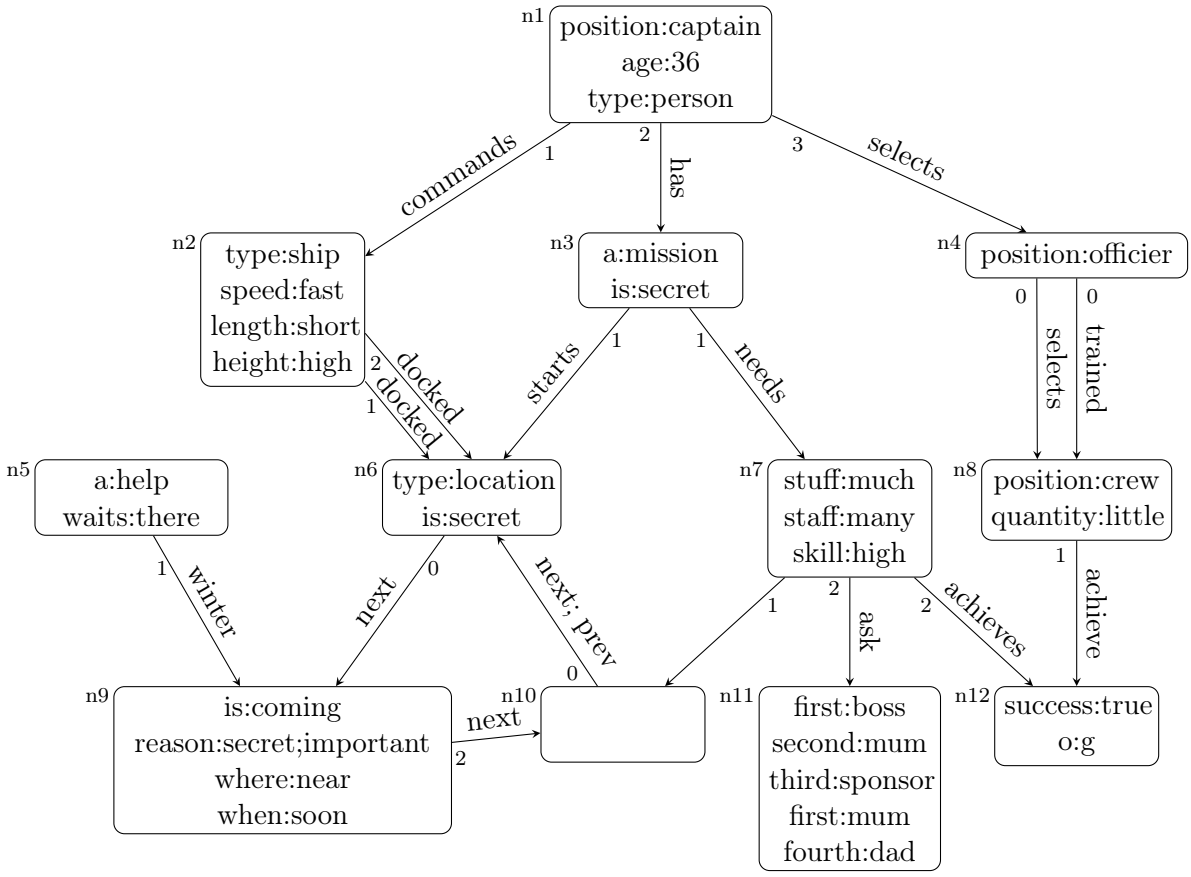


Figure 3: Task specification for Exercise 2: Example graph to be checked by the EER diagram in Figure 2

Exercise 3 (Construct a relational schema) [2 points]

Construct a relational schema according to the EER-diagram given in Figure 4. `NULL` values are not allowed (you can assume that all attributes specified for an entity type exist for all entities of this type, i.e., the definedness of all attributes is 100%). Create as few relations as possible without introducing any redundancies. For each relation clearly mark the primary keys by underlining the corresponding attributes and display foreign keys in italics.

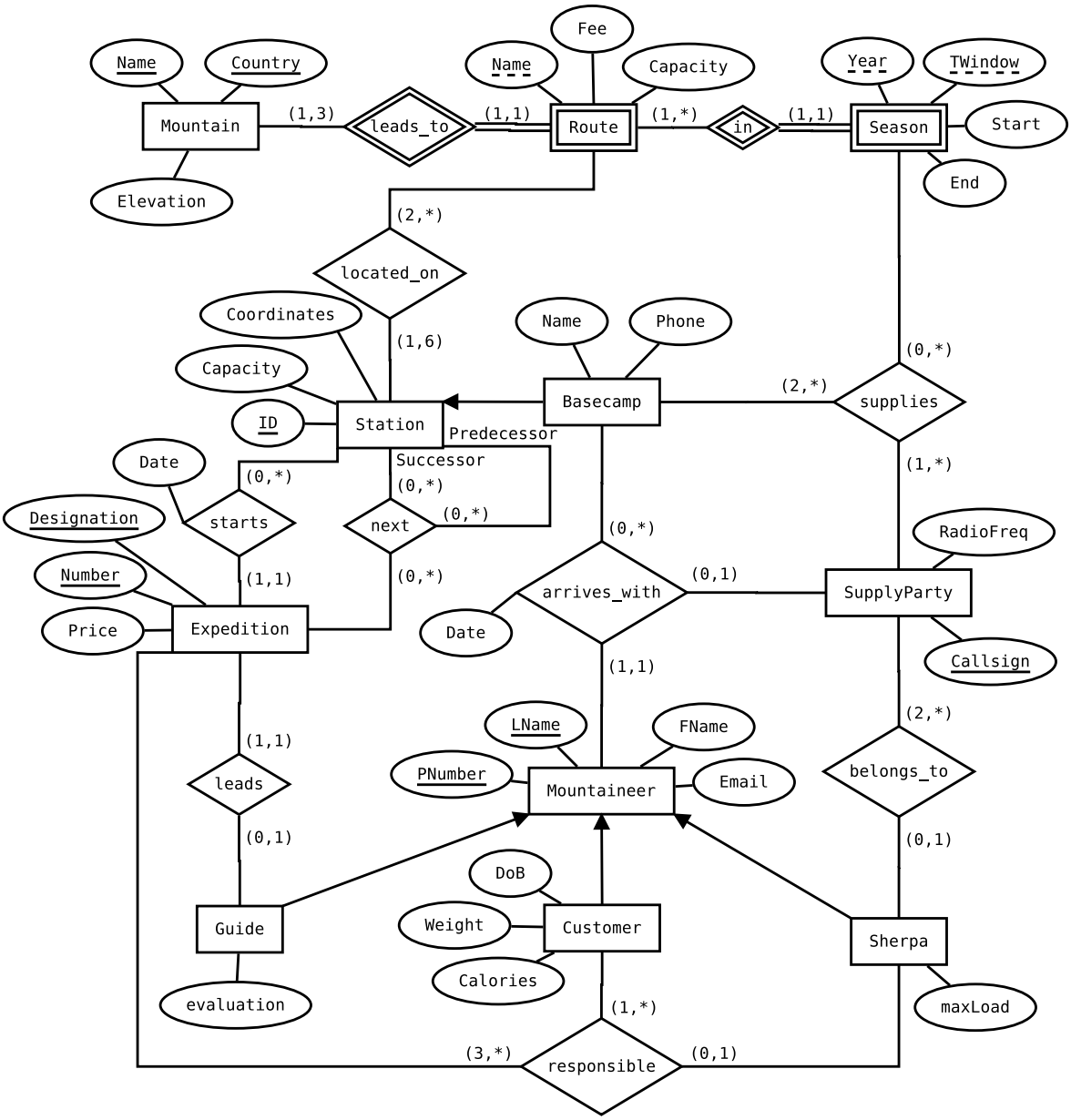


Figure 4: EER-diagram for exercise 3

Solution:

```
Mountain     (name, country, elevation)
Route        (mountain_name: Mountain.name, country: Mountain.country,
              route_name, fee, capacity)
Season       (mountain_name: Route.mountain_name, country: Route.country,
              route_name: Route.route_name, year, twindow, start, end)
Station      (ID, coordinates, capacity)
Basecamp     (ID:Station.ID, name, phone)
Expedition   (designation, number, price, ID: Station.ID, date,
              pnumber: Guid.pnumber, lname: Guide.lname)
Mountaineer (pnumber, lname, fname, email, date,
              basis_id: Basecamp.ID, callsign: SupplyParty.callsign)
Guide        (pnumber: Mountaineer.pnumber, lname: Mountaineer.lname,
              evaluation)
Customer     (pnumber: Mountaineer.pnumber, lname: Mountaineer.lname
              dob, weight, calories)
Sherpa       (pnumber: Mountaineer.pnumber, lname: Mountaineer.lname,
              maxload)
SupplyParty (callsign, radiofreq)
loacted_on   (Mountain_name: Route.mountain_name, country: Route.country,
              ID: Station.ID)
next         (predecessor_id: Station.ID, successor_id: Station.ID,
              designation: Expedition.designation,
              number: Expedition.number)
responsible (sherpa_pnumber: Sherpa.pnumber,
              sherpa_lname: Sherpa.lname,
              customer_pnumber: Customer.pnumber,
              customer_lname: Customer.lname,
              designation: Expedition.designation,
              number: Expedition.number)
belongs_to   (sherpa_pnumber: Sherpa.pnumber,
              sherpa_lname: Sherpa.lname,
              callsign: SupplyParty.callsign)
supplies     (camp_id: Basecamp.ID, callsign: SupplyParty.callsign
              mountain_name: Season.mountain_name, country: Season.country,
              route_name: Season.route_name, year: Season.year,
              twindow: Season.twindow)
```

Exercises: Relational Algebra - Relational Calculus

To help with typesetting the solutions to the following exercises, we compiled a list of the most important symbols for the Relational Algebra at `http://dbai.tuwien.ac.at/education/dm/resources/symbols.html`. You can copy and paste them into your Word/ LibreOffice/ OpenOffice/... document. In addition, the corresponding LaTeX commands are listed as well.

Exercise 4 (Evaluation) [0.4 points]

Consider the four relations below.

Drug		
name	**manufacturer**	**substance**
Hairloss	TU	Exam
Tiredness	Uni	Task
Success	TU	Knowledge

Box		
name	**boxsize**	**price**
Hairloss	2	19
Hairloss	5	20
Tiredness	1	5
Success	42	18, 70

Branch-office	
address	**bmanager**
FH	H.A.S. Ended
GM	I.S. Done
Trei	R.E. Used
Fav	F. Inished
EI	C.O. Pied

instore			
address	**name**	**boxsize**	**amount**
Fav	Tiredness	1	5
GM	Hairloss	5	7
FH	Hairloss	2	13
Fav	Success	42	1
Trei	Tiredness	1	8

Provide the results of the following queries over these relations.

(a)

$$\pi_{\text{substance,bmanager}}\Big(\pi_{\text{substance}}(\sigma_{\text{manufacturer}=\text{“Uni”}}(\texttt{Drug}))\times \big((\sigma_{\text{boxsize}<2}(\texttt{Box}) \bowtie \sigma_{\text{amount}>6}(\texttt{instore})) \rtimes \texttt{branch-office}\big)\Big)$$

Solution:

q	
substance	**bmanager**
Task	R.E. Used

(b)

$$\{f \mid f \in \texttt{branch-office} \wedge \forall l \in \texttt{instore}(l.\text{address} = f.\text{address} \rightarrow \neg(\exists m \in \texttt{Drug}(m.\text{name} = l.\text{name} \wedge m.\text{manufacturer} \neq \text{“TU”})))\}$$

Solution:

q	
adresse	**fleiter**
FH	H.A.S. Ended
GM	F. Inished
EI	C.O. Pied

Exercise 5 (Equivalences) [2 points]

Consider the following pairs q_i, q_j of expressions in Relational Algebra over the relational schemas $R(\underline{A}BC)$, $S(B\underline{D}E)$ and $T(\underline{A}DF)$.

- Verify, whether the two expressions are equivalent (i.e. whether they produce the same result on all possible instances of the relational schemas). You can assume that no `NULL`-values may occur in any instance of the schemas.
- Justify your answer with a brief **explanation**.
- In case the two expressions are *not* equivalent, additionally provide a **counterexample**. (A counterexample consists of the concrete instances of the affected relational schemas and the results of both expressions over these instances.)
  In case one of the expressions is not a valid expression in Relational Algebra you do not have to provide a counterexample, hence in this case an explanation suffices.

(a) $q_1\colon \big(\sigma_{A=D}(R \bowtie S)\big) \ltimes T$ and
$q_2\colon (R \bowtie_{R.A=S.D \wedge R.B=S.B} S) \ltimes \sigma_{A=D}(T)$

(b) $q_3\colon \pi_{AD}\big(\big((\pi_B(R) - \pi_B(S)) \bowtie R\big) \bowtie S\big) \cap \pi_{AD}(T)$ and
$q_4\colon \pi_{AD}(T) - \rho_{D \leftarrow B}\big(\pi_{AB}(R \bowtie S) \cup \rho_{B \leftarrow D}(\pi_{AD}(T))\big)$

(c) $q_5\colon R \bowtie \big(\pi_{AF}(T) \cap \rho_{A \leftarrow E, F \leftarrow D}(\pi_{DE}(S))\big)$ and
$q_6\colon (R \bowtie \pi_{AF}(T)) \cap (R \bowtie \rho_{A \leftarrow D, F \leftarrow E}(\pi_{DE}(S)))$

(d) $q_7\colon \rho_{A \leftarrow Q.A, B \leftarrow Q.B}\big(\pi_{Q.A,Q.B}\big(\sigma_\theta(\rho_Q(\pi_A(R) \times \pi_B(R)) \times (R \times T))\big)\big)$
where $\theta = (Q.A = R.A \wedge Q.B = R.B) \vee (Q.A = T.A \wedge Q.B = T.F)$ and
$q_8\colon \big(\pi_{AB}(R) \cup \rho_{B \leftarrow F}(\pi_{AF}(T))\big)$

Solution:

Exercise (a) **No**, q_1 and q_2 **are not** equivalent.

The relations defined by the two queries have different schemas! The schema of the result of q_1 is $q_1(A, B, C, D, E)$, while the schema of the result of q_2, being $q_2(A, R.B, C, S.B, D, E)$, contains an additional attribute.

The reason for this is that the natural join only keeps one copy of each attribute occurring in the schema of both, R and S, while the θ-join keeps both.

Counterexample

R		
A	**B**	**C**
1	2	1

S		
B	**D**	**E**
2	1	1

q_1				
A	**B**	**C**	**D**	**E**
1	2	1	1	1

q_2					
A	**R.B**	**C**	**S.B**	**D**	**E**
1	2	1	2	1	1

Exercise (b) **Yes**, q_3 and q_4 **are** equivalent.

Over all possible instances, both queries return an empty relation. In addition, these empty relations agree on their schemas.

Exercise (c) **No**, q_5 and q_6 **are not** equivalent.

The query q_5 "switches" the attributes of S. I.e., while in q_6 the attribute D of S corresponds to the attribute A int T and R and the attribute E in S corresponds to the attribute F in T, in the query q_5 the attribute D of S is compared to F in T, and the attribute E of S to the attribute A in R and T. This returns in different results as soon as there are tuples that agree in one case, but not in the other. One such example is given below.

Counterexample

R		
A	**B**	**C**
2	1	1

S		
B	**D**	**E**
1	2	3

T		
A	**D**	**F**
2	1	3

q_5			
A	**B**	**C**	**F**

q_6			
A	**B**	**C**	**F**
2	1	1	3

Exercise (d) **No**, q_7 and q_8 **are not** equivalent.

The query q_8 returns the union of the tuples in $\pi_{AB}(R)$ and $\pi_{AF}(T)$, with the attribute F in T being renamed to B for the two relations to be compatible.

The reason why q_7 is not able to always return the same result is that an inspection of q_7 reveals that it only returns values occurring in R, while, as discussed above, q_8 returns values that occur in either R or T.

Counterexample

R		
A	**B**	**C**
1	2	1

T		
A	**D**	**F**
a	b	c

q_7	
A	**B**
1	2

q_8	
A	**B**
1	2
a	c

Exercise 6 (Answer Sizes) [2 points]

Consider the relational schemas $R(\underline{AB})$, $S(AB\underline{CD})$, and $T(AC\underline{E})$ and an instance of every schema, where there are $|R|$ tuples in the instance of R, $|S|$ tuples in the instance of S, and $|T|$ tuples in the instance of T.

- Provide the minimal and maximal size (= number of tuples) of the following expressions in Relational Algebra for the given values of $|R|$, $|S|$, $|T|$.
- Justify your answer.

- For both, the smallest and biggest possible answer size, provide concrete instances of the schemas (with R, S, and T having $|R|$, $|S|$, and $|T|$ tuples, respectively) over which the query returns an answer with the minimal/maximal number of tuples.

(a) q_1: $(\pi_A(R) \cup \pi_A(T)) \cup (\pi_A(R) \cap \pi_A(T))$ (where $|R| = 4$ and $|T| = 7$)

(b) q_2: $\big((\sigma_{C=4 \wedge D=2}(S) \bowtie S) \rtimes T\big) \bowtie T$ (where $|S| = 5$ and $|T| = 7$)

(c) q_3: $\rho_{F \leftarrow B}(R) \bowtie \sigma_{A=4 \wedge B=2}(S)$ (where $|R| = 3$ and $|S| = 2$)

(d) q_4: $\pi_{AF}(\rho_{F \leftarrow A}(T) \times R) - \pi_{AF}(\sigma_{F>A}(\rho_{F \leftarrow A}(S) \bowtie R))$ (where $|R| = 2$, $|S| = 3$ and $|T| = 3$)

Solution:

Exercise (a) [Minimum: 4 | Maximum: 11]

Since A is the key of R, the subquery $\pi_A(R)$ contains 4 different tuples (since each of the four tuples in R contains a different value for A). Since A is no key in T, the result of the subquery $\pi_A(T)$ may contain between 1 and 7 tuples. For $\pi_A(R) \cup \pi_A(T)$ we thus get that the result mayu contain between 4 (all tuples in $\pi_A(T)$ also appear in $\pi_A(R)$) and 11 (all tuples in the two result sets are distinct) tuples.

The answer to the subquery $\pi_A(R) \cap \pi_A(T)$ is always a subset of the answer to the subquery $\pi_A(R) \cup \pi_A(T)$, and thus it does not alter the result.

Minimum: 4

R	
A	**B**
1	1
2	2
3	3
4	4

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

Result
A
1
2
3
4

Maximum: 11

R	
A	**B**
1	1
2	2
3	3
4	4

T		
A	**C**	**E**
5	1	1
6	1	2
7	1	3
8	1	4
9	1	5
a	1	6
b	1	7

Result
A
1
2
3
4
5
6
7
8
9
a
b

Exercise (b) [Minimum: 7 | Maximum: 7]

Both, the upper and the lower bound follow from the following observations:

The result of the subquery

$$(\sigma_{C=4 \wedge D=2}(S) \bowtie S) \rtimes T$$

is either T or a relation containing a subset of the tuples in T. The schema of the result of the subquery is exactly the schema of T.

Just like the natural join, all variants of the outer join keep only one copy of each attribute occurring in both schemas in the resulting schema. Therefore also the schema of the result of q_2 is identical to the schema of T.

Because of this, the result of q_2 is *always* the relation T: Since the result of

$$(\sigma_{C=4 \wedge D=2}(S) \bowtie S) \rtimes T$$

can only contain tuples from T, each tuple in T can find at most one join partner. At the same time, also tuples who do not find a join partner occur in the result exactly once (by padding the additional attributes with `NULL`). Since in this case, the schema of the result is identical to the schema of T, this means that no `NULL`s need to be added, and each tuple of T appears in the result: Either, because the tuple is also part of the result of the subquery on the left hand side of the $\bowtie$, or because it is in the result without having a join partner.

Minimum: 7

S			
A	**B**	**C**	**D**
1	1	0	0
1	1	3	2
1	1	2	2
1	1	1	2
1	1	5	2

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

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

Maximum: 7

S			
A	**B**	**C**	**D**
1	1	4	2
1	1	3	2
1	1	2	2
1	1	1	2
1	1	5	2

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

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

Exercise (c) [Minimum: 3 | Maximum: 5]

The result of

$$\rho_{F \leftarrow B}(R)$$

always contains the three tuple from R (only under a different schema). As a result, the full outer join

$$\rho_{F \leftarrow B}(R) \mathbin{⟗} \sigma_{A=4 \wedge B=2}(S)$$

always contains at least three tuples, since the three tuples from $\rho_{F \leftarrow B}(R)$ will be part of the result (perhaps padded with `NULL` values). The overall result thus always contains at least three tuples. This minimal size can also be actually reached. In fact, this may happen in several different ways. One possibility is that

$$\sigma_{A=4 \wedge B=2}(S)$$

returns an empty result set.

For the upper bound, we first observe that the biggest possible result for

$$\sigma_{A=4 \wedge B=2}(S)$$

contains two tuples, since AB is no key of S. AS a next step, we observe what comprises the biggest possible result of the outer join. For this, we have to check two different possibilities. The first way to generate a maximal size result is by combining all tuples on the left hand side with all tuples on the right hand side to create a result tuple. However, since the shared attribute between the relations that implements the join is a key of R, this approach is not feasible, since it would require all tuples to contain the same value on A. The alternative approach is thus to combine one tuple on the left hand side with two tuples on the right hand side to two tuples, and then to pad the remaining tuples with `NULL` values. This produces four tuples.

However, this is not the biggest possible solution. This is created by the results of the left- and right hand side having no tuples with matching values for A. In this case, the result contains all tuples from the left hand side and all tuples from the right hand side, all padded with `NULL` values. This thus gives 5 tuples.

Minimum: 3

R	
A	**B**
1	1
2	2
4	4

S			
A	**B**	**C**	**D**
4	2	1	1
2	2	2	2

Result				
A	**F**	**B**	**C**	**D**
1	1	NULL	NULL	NULL
2	2	NULL	NULL	NULL
4	4	2	1	1

Minimum: 5

R	
A	**B**
1	1
2	2
3	3

S			
A	**B**	**C**	**D**
4	2	1	1
4	2	2	2

Result				
A	**F**	**B**	**C**	**D**
1	1	NULL	NULL	NULL
2	2	NULL	NULL	NULL
3	3	NULL	NULL	NULL
4	NULL	2	1	1
4	NULL	2	2	2

Exercise (d) [Minimum: 0 | Maximum: 6]

For the lower bound it is sufficient to show that the result of the subquery on the right hand side of the "difference" operator can be contained in the result of the left hand side. One such case is given in the example below.

The upper bound is produced when the result of the left hand side of the "difference" operator is as big as possible, and as little as possible of these tuples occur in the result of the subquery on the right hand side.

Now the results of

$$\rho_{F \leftarrow A}(T) \times R$$

always contains six tuples. Since R cannot contain any two tuples that share the same value for A, and there is no reason against all three tuples in T having different values on A, all these tuples may have different values on AF, leading to the biggest possible result of the subquery on the left hand side containing six tuples.

At the same time, the result of the subquery on the right hand side may be empty for several reasons. In all these cases, all six tuples remain in the reslt of the query.

Minimum: 0

R	
A	**B**
1	1
2	1

S			
A	**B**	**C**	**D**
7	1	1	1
8	1	2	2
9	1	3	3

T		
A	**C**	**E**
7	4	1
7	4	2
7	4	3

Result	
A	**F**

Maximum: 6

R	
A	**B**
1	1
2	2

S			
A	**B**	**C**	**D**
0	1	0	0
7	7	7	7
8	8	8	8

T		
A	**C**	**E**
3	3	3
4	4	4
5	5	5

Result	
A	**F**
1	3
1	4
1	5
2	3
2	4
2	5

Exercise 7 (Primitive Operations) [1 point]

(a) Express the operator $\bowtie$ using only the primitive operators of Relational Algebra.

To achieve that, use two relations, R and S. Schema R contains the attributes $(R_1, \ldots, R_r, G_1, \ldots, G_g)$ and schema S contains the attributes $(S_1, \ldots, S_s, G_1, \ldots, G_g)$. Express $R \bowtie S$ using primitive operators only.

Hint: Your expression may also contain constant relations. Keep in mind that r, s and g are not constants. A constant relation is for instance: $\{(1)\}$.

Solution:
In a first step, we divide the operator $\bowtie$ into two compontents: One component (a natural join) takes care that all tuples that have join partners are combined with them to form new tuples in the results.

The second part selects the tuples without join partnes and combines them with the number of of `NULL`s necessary to form a tuple of equired arity.

$$R \bowtie S = (R \bowtie S) \cup \big((\ldots((R - \pi_{R_1,\ldots,R_r,G_1,\ldots G_g}(R \bowtie S)) \times \rho_{S_1 \leftarrow N}(N)) \times \cdots \times \rho_{S_s \leftarrow N}(N))\big),$$

where N denotes the constant relation with the schema $N(N)$ and the instance $\{(\texttt{NULL})\}$.

Since the natural join $\bowtie$ is not a primitive operator, it also needs to be translated into an expression using only primitive operators. Thus both occurrences of the expression $R \bowtie S$ need to be replaced by

$$\pi_{R_1,\ldots,R_r,G_1,\ldots,G_g,S_1,\ldots,S_s}(\sigma_{\bigwedge_{i=1}^{g} G_i = H_i}(R \times \rho_{H_1 \leftarrow G_1,\ldots,H_g \leftarrow G_g}(S))).$$

(b) Translate the query $R \ltimes S$ for the relations R and S with the schemas from exercise (a) into both, tuple calculus and domain calculus ("Domänenkalkül").

Solution:
Tuple calculus:

$$\{t \mid t \in R \wedge \exists u \in S(\bigwedge_{i=1}^{g} t.G_i = u.G_i)\}$$

Domain calculus:

$$\{[u_1,\ldots,u_r,v_1,\ldots,v_g] \mid [u_1,\ldots,u_r,v_1,\ldots,v_g] \in R \wedge \exists w_1,\ldots,w_s([w_1,\ldots,w_s,v_1,\ldots,v_g] \in S)\}$$

(c) Translate the following query over the relations R, S and T first into a query in Relational Algebra which uses primitive operators only, and then into the tuple- and domain calculus. R and S have the same schema, as in the previous exercise (a) and T has the schema $(T_1,\ldots T_t, G_1,\ldots G_g)$.

$$\pi_{T_1,\ldots,T_t,G_1,\ldots G_g}(T \bowtie (\pi_{G_1,\ldots,G_g}(R) - \pi_{G_1,\ldots,G_g}(S)))$$

Solution:
Primitive Operators:

$$\pi_{T.T_1,\ldots,T.T_t,T.G_1,\ldots T.G_g}\big(\sigma_{T.G_1=R.G_1 \wedge \cdots \wedge T.G_g=R.G_g}(T \times \rho_R(\pi_{G_1,\ldots,G_g}(R) - \pi_{G_1,\ldots,G_g}(S)))\big)$$

Tuple Calculus:

$$\{w \mid w \in T \wedge \exists u \in R(\bigwedge_{i=1}^{g} u.g_i = w.g_i \wedge \neg(\exists v \in S(\bigwedge_{i=1}^{g} v.g_i = w.g_i)))\}$$

Domain Calculus:

$$\{[w_1,\ldots,w_t,a_1,\ldots,a_g] \mid [w_1,\ldots,w_t,a_1,\ldots,a_g] \in T \wedge \exists u_1,\ldots,u_r([u_1,\ldots,u_r,a_1,\ldots,a_g] \in R) \wedge \neg(\exists v_1,\ldots,v_s([v_1,\ldots,v_s,a_1,\ldots,a_g] \in S))\}$$

Exercise 8 (Formalizing Queries) [3.6 points]

A cafe manages its sales data with the following schema (Primary keys are underlined, foreign keys are written in *italics*).

```
Barista     (FName, SName, BirthDat, Salary)
learned_from(TeacherFName: Barista.FName, TeacherSName: Barista.SName,
             StudentFName: Barista.FName, StudentSName: Barista.SName)
Customer    (CNr, Email)
Coffee      (Designation, Description, Price, Difficulty)
Purchase    (Date, CNr: Customer.CNr, Coffee: Coffee.Designation,
             FName: Barista.FName, SName: Barista.SName, Amount, Rating)
Beans       (Name, Country-of-origin, Aroma, Quality)
contains    (Coffee: Coffee.Designation,
             BeansN: Beans.Name, BeansC: Beans.Country-of-origin, Amount)
```

(In the following you may use suitable (unique) abbreviations for relations and table names.)

Express the queries described below in **Relational Algebra**, the **tuple calculus** (Tupelkalkül) and the **domain calculus** (Domänenkalkül).

Exercises marked with **, i.e. **Exercises (a), (b), (c) and (h)**, are *not mandatory*. You will not get any points for solving them. Exercises (a), (b) and (c) are simple exercises, in case you do not have any experience with the formulation of queries in Relational Algebra (or database queries in general) yet. Exercise (h) is more complex, in case you want another example to practice. You find the solutions for all non-mandatory exercises at the end of this document.

(a) ** Find all Barista (FName, SName, BirthDat) who earn more than 2500.

Solution:
Relational Algebra:

$$\pi_{\text{FName,SName,BirthDat}}(\sigma_{\text{Salary}>2500}(\texttt{Barista}))$$

Tuple Calculus:

$$\{[b.\text{FName}, b.\text{SName}, b.\text{BirthDat}] \mid b \in \texttt{Barista} \wedge b.\text{Salary} > 2500\}$$

Domain Calculus:

$$\{[fn, sn, bdat] \mid \exists s([fn, sn, bdat, s] \in \texttt{Barista} \wedge s > 2500)\}$$

(b) ** Find all Barista (FName, SName, BirthDat) that have never received a 5-star rating (or better). Assume that the rating is stored as natural number.

Solution:
Relational Algebra:

$$\pi_{\text{FName,SName,BirthDat}}\big(\texttt{Barista} \ltimes \big(\pi_{\text{FName,SName}}(\texttt{Barista}) - \pi_{\text{FName,SName}}(\sigma_{\text{Rating}\geq 5}(\texttt{Purchase}))\big)\big)$$

Tuple Calculus:

$$\{[b.\text{FName}, b.\text{SName}, b.\text{BirthDat}] \mid b \in \texttt{Barista} \wedge \neg(\exists k \in \texttt{Purchase}(k.\text{SName} = b.\text{FName} \wedge k.\text{SName} = b.\text{SName} \wedge k.\text{Rating} \geq 5))\}$$

Domain Calculus:

$$\{[fn, sn, bdat] \mid \exists s([fn, sn, bdat, s] \in \texttt{Barista} \wedge \neg(\exists d, cnr, cof, a, r([d, cnr, cof, fn, sn, bdat, a, r] \in \texttt{Purchase} \wedge r \geq 5)))\}$$

(c) ** For the customer with CNr 74656, find all countries from where the beans of her purchases between 01.10.2371 and 20.12.2371 originate. Assume that the relation `contains` only contains entries with values greater than 0 for the `Amount`-attribute, i.e. only entries for beans which are used in coffee are stored.

Solution:
Relational Algebra:

$$\pi_{\text{BeansC}}(\sigma_{\text{Date}\geq 01.10.2371 \wedge \text{Date}\leq 20.12.2371 \wedge \text{CNr}=74656}(\rho_{\text{Numbers}\leftarrow\text{Amount}}(\texttt{Purchase})) \bowtie \texttt{contains})$$

Tuple Calculus:

$$\{[c.\text{BeansC}] \mid c \in \texttt{contains} \wedge \exists p \in \texttt{Purchase}(p.\text{Coffee} = c.\text{Coffee} \wedge p.\text{CNr} = 74656 \wedge p.\text{date} \geq 01.10.2371 \wedge p.\text{date} \leq 20.12.2371)\}$$

Domain Calculus:

$$\{[o] \mid \exists c, b, a, d, fn, sn, a2, r([c, b, o, a] \in \texttt{contains} \wedge [d, 74656, c, fn, sn, a2, r] \in \texttt{Purchase} \wedge d \geq 01.10.2371 \wedge d \leq 20.12.2371)\}$$

(d) Find all Barista (list the attribute SName only) with a salary less than 1800, who have already received a 3-star rating (or better) for a coffee with the difficulty 5. Assume that both, the attribute `Rating` and `Difficulty` are stored as natural numbers.

Solution:
Relational Algebra:

$$\pi_{\text{SName}}\big(\sigma_{\text{Salary}<1800}(\texttt{Barista}) \bowtie \sigma_{\text{Difficulty}=5 \wedge \text{Rating}\geq 3}(\rho_{\text{Coffee}\leftarrow\text{Designation}}(\texttt{Coffee}) \bowtie \texttt{Purchase})\big)$$

Tuple Calculus:

$$\{[purchase.\text{SName}] \mid purchase \in \texttt{Purchase} \wedge purchase.\text{Rating} \geq 3 \wedge \exists k \in \texttt{Coffee}(k.\text{Designation} = purchase.\text{Coffee} \wedge k.\text{Difficulty} = 5 \wedge \exists b \in \texttt{Barista}(b.\text{FName} = purchase.\text{FName} \wedge b.\text{SName} = purchase.\text{SName} \wedge b.\text{Salary} < 1800))\}$$

Domain Calculus:

$$\{[sn] \mid \exists k, d, cnr, fn, m, b, t, p, dob, g([k, t, p, 5] \in \texttt{Coffee} \wedge [fn, sn, dob, g] \in \texttt{Barista} \wedge [d, cnr, k, fn, sn, m, b] \in \texttt{Purchase} \wedge b \geq 3 \wedge g < 1800)\}$$

(e) Find the coffees (attribute `Designation` only) which contain beans from all countries stored in the database.

Solution:

Relational Algebra:

$$\pi_{\text{Coffee,BeansC}}(\texttt{contains}) \div \rho_{\text{BeansC} \leftarrow \text{CountryOfOrigin}}(\pi_{\text{CountryOfOrigin}}(\texttt{Beans}))$$

Tuple Calculus:

$$\{[k.\text{Designation}] \mid k \in \texttt{Coffee} \wedge \forall b \in \texttt{Beans}(\exists e \in \texttt{contains}(e.\text{Coffee} = k.\text{Designation} \wedge e.\text{BeansC} = b.\text{CountryOfOrigin}))\}$$

Domain Calculus:

$$\{[k] \mid \exists txt, pr, lvl\big([k, txt, pr, lvl] \in \texttt{Coffee} \wedge \forall b, country, ar, qu([b, country, ar, qu] \in \texttt{Beans} \rightarrow \exists bn, m([k, bn, country, m] \in \texttt{contains}))\big)\}$$

(f) For all Barista find the countries (you can use countries that are stored in the database only) from which beans originate, that have never been used in a coffee sold by the Barista before. Provide a table with the columns (SName, Land), where "SName" is the Surname of the Barista and "Land" the name of the origin country.

Solution:
Relational Algebra:

$$\pi_{\text{SName, BeansC}}(\rho_{\text{BeansC} \leftarrow \text{CountryOfOrigin}}(\pi_{\text{FName,SName,CountryOfOrigin}}(\texttt{Barista} \times \texttt{Beans})) - \pi_{\text{FName,SName,BeansC}}(\rho_{\text{Anzahl} \leftarrow \text{Menge}}(\texttt{Purchase}) \bowtie \texttt{contains}))$$

Tuple Calculus:

$$\{[ba.\text{SName}, bo.\texttt{CountryOfOrigin}] \mid ba \in \texttt{Barista} \wedge bo \in \texttt{Beans} \wedge \neg\big(\exists e \in \texttt{contains}(e.\text{BeansC} = bo.\text{CountryOfOrigin} \wedge \exists k \in \texttt{Purchase}(k.\text{FName} = ba.\text{FName} \wedge k.\text{SName} = ba.\text{SName} \wedge k.\text{Coffee} = e.\text{Coffee}))\big)\}$$

Domain Calculus:

$$\{[sn, country] \mid \exists fn, dob, sal, bo, ar, qual([fn, sn, dob, sal] \in \texttt{Barista} \wedge [bo, country, ar, qual] \in \texttt{Beans} \wedge \neg(\exists dat, cnr, coff, am, rat, bam, bnam([dat, cnr, coff, fn, sn, am, rat] \in \texttt{Purchase} \wedge [coff, bnam, country, bam] \in \texttt{contains})))\}$$

(g) Find the names of all beans that have been used in any coffee with difficulty 3 or higher, sold on 1.10.2019.

Solution:
Relational Algebra:

$$\pi_{\text{BeansN}}\big(\pi_{\text{BeansN,BeansC}}(\rho_{\text{Count}\leftarrow\text{Amount}}(\sigma_{\texttt{Date}=1.10.2019}(\texttt{Purchase})) \bowtie \texttt{contains}) \cap \pi_{\text{BeansN,BeansC}}(\sigma_{\text{Difficulty}\geq 3}(\texttt{Coffee}) \bowtie \texttt{contains})\big)$$

Tuple Calculus:

$$\{[b.\text{Name}] \mid b \in \texttt{Beans} \wedge \exists purchase \in \texttt{Purchase}(purchase.\text{Date} = 1.10.2019 \wedge \exists e \in \texttt{contains}(e.\text{Coffee} = purchase.\text{Coffee} \wedge e.\text{BeansN} = b.\text{BeansN} \wedge e.\text{BeansC} = b.\text{BeansC} \wedge \exists coff \in \texttt{Coffee}(coff.\text{Difficulty} \geq 3 \wedge \exists cont \in \texttt{contains}(cont.\text{Coffee} = coff.\text{Designation} \wedge cont.\text{BeansN} = b.\text{BeansN} \wedge cont.\text{BeansC} = b.\text{BeansC})))\}$$

Domain Calculus:

$$\{[beans] \mid \exists country, ar, qual, cnr, coffc, fn, sn, numb, rat, amount1, amount2, coffs, txt, pr, difc([beans, country, ar, qual] \in \texttt{Beans} \wedge [coffc, beans, country, amount1] \in \texttt{contains} \wedge [1.10.2019, cnr, coffc, fn, sn, numb, rat] \in \texttt{Purchase} \wedge [coffs, beans, country, amount2] \in \texttt{contains} \wedge [coffs, txt, pr, difc] \in \texttt{Coffee} \wedge difc \geq 3)\}$$

(h) ** First, consider all teachers who have at least one student that earns more than 5000. Then for the solution find the list of ratings for all students of these teachers.

Solution:
Relationa Algebra:

$$\pi_{\texttt{Purchase}.\text{FName},\text{Rating}}\big(\sigma_{\phi}(\sigma_{\text{Salary}\geq 5000}(\texttt{Barista})\times\rho_{s1}(\texttt{learned\_from})\times\rho_{s2}(\texttt{learned\_from})\times\texttt{Purchase})\big)$$

where ϕ is the following condition:

$$\begin{aligned}\phi = &\texttt{s1}.\text{StudentFName} = \text{FName} \wedge \texttt{s1}.\text{StudentSName} = \text{SName} \wedge\\ &\texttt{s1}.\text{TeacherFName} = \texttt{s2}.\text{TeacherFName} \wedge \texttt{s1}.\text{TeacherSName} = \texttt{s2}.\text{TeacherSName} \wedge\\ &\texttt{s2}.\text{StudentFName} = \texttt{Purchase}.\text{FName} \wedge \texttt{s2}.\text{StudentSName} = \texttt{Purchase}.\text{SName}\end{aligned}$$

Tuple Calculus:

$$\begin{aligned}\{[s.\text{FName}, k.\text{Rating}] \mid\ & s \in \texttt{learned\_from} \wedge p \in \texttt{Purchase} \wedge\\ & s.\text{StudentFN} = p.\text{FName} \wedge s.\text{StudentSName} = p.\text{SName} \wedge\\ & \exists l \in \texttt{learned\_from}(\\ & \quad l.\text{TeacherFName} = s.\text{TeacherFN} \wedge l.\text{TeacherSN} = s.\text{TeacherSN} \wedge\\ & \quad \exists b \in \texttt{Barista}(\\ & \quad\quad b.\text{FN} = l.\text{StudentFN} \wedge\\ & \quad\quad b.\text{SN} = l.\text{StudentSN} \wedge\\ & \quad\quad b.\text{Salary} \geq 5000))\}\end{aligned}$$

Domain Calculus:

$$\begin{aligned}\{[svn, rat] \mid \exists bbn, bsn, bdat, sal([bfn, bsn, bdat, sal] \in \texttt{Barista} \wedge sal \geq 5000 \wedge\\ \exists lfn, lsn([lfn, lsn, bfn, bsn] \in \texttt{learned\_from} \wedge\\ \exists ssn([lfn, lsn, sfn, ssn] \in \texttt{learned\_from} \wedge\\ \exists dat, cnr, cof, amo(\\ [dat, cnr, cof, sfn, ssn, amo, rat] \in \texttt{Purchase})))\}\end{aligned}$$