1 Constructing an ER Model

The detailed description of a database to store information about a ski resort is given below:

The database stores the first and last name of persons, who are identified by their social security number (SSN). Every person is either a customer or an instructor (but not both). A customer can have many credit cards. Every instructor has a certificate. A certificate has a description and is identified by its title. Each instructor teaches one or more one-day courses and each one-day course is taught by exactly one instructor. A one-day course is identified by its name, takes place on a specific date and has a number of participating customers. A customer that participates in a one-day course pays a certain price (can vary between customers) and is awarded with a trophy. The database stores the sold season tickets, which are identified by a combination of the date when they are issued and an incrementing counter (e.g. the fifth ticket on 14.04.17). Moreover, the price of a season ticket is stored. A season ticket grants free participation in one one-day course. A customer can own several season tickets.

Tasks:

1. Draw an ER model for the given database description. Make sure to include keys, cardinalities, and participation constraints.
The previous ER-model does not allow us to model recurring courses, e.g. we cannot have a course with Name “Ski Trial Course” in 2016 and the same course in 2017, because Name is the sole primary key. Pushing attribute Date into a composite primary key (e.g. “Ski Trial Course 2016”) is also not ideal, because we then introduce redundancy. Imagine a course had many other non time-dependent attributes, e.g. a description, a syllabus etc. Then we would have to repeat this information for each year the course takes place. Another approach that alleviates the described problem is shown in the following ER-model. The difference is that we pushed the Date attribute from entity One-Day Course into its relationships. Therefore we can have a “Ski Trial Course”, which is independent of any specific date, and when e.g. a customer takes this course, we record its specific date in the relationship. In this model, the derived attribute #Participants then has different semantics; it now represents the number of participants that this course attracted since its conception.

2. For each relationship type in the resulting ER model explain your choice of the cardinality and participation constraint. An explanation should consist of references to the description and assumptions, which do not contradict the database description.

For example:

(a) \{Instructor:Certificate = M:N\}

Reference: “Every instructor has a certificate”

Assumption: Many instructors can have the same certificate. There is no instructor without a certificate (thus total participation).
Solution:

(b) \{\text{Instructor:One-Day Course} = 1:N\}
   \text{Reference: “Each instructor teaches one or more one-day courses and}
   \text{each one-day course is taught by exactly one instructor.”}

(c) \{\text{Customer:One-Day-Course} = M:N\}
   \text{Reference: “A customer that participates in a one-day course pays}
   \text{a certain price (can vary between customers) and is awarded with a}
   \text{trophy.”}
   \text{Assumptions: A one-day course can host more than one customer.}
   \text{A customer can attend more than one one-day course.}

(d) \{\text{Customer:Season Ticket} = 1:N\}
   \text{Reference: “A customer can own several season tickets.”}
   \text{Assumptions: A season ticket must belong to exactly one customer.}

(e) \{\text{Season Ticket:One-Day Course} = N:1\}
   \text{Reference: “A season ticket grants free participation in one one-day}
   \text{course.”}
   \text{Assumption: A one-day course can host several customers with sea-
   \text{son tickets. A season ticket must grant free participation in one}
   \text{one-day course.}

2 Converting an ER Model

Map the following ER model, including primary and foreign keys, to a corresponding relational database schema. Avoid NULL values and redundancy as much as possible.

\[
\begin{align*}
&V1 & V2 \\
&V3 & & V4 \\
& & V5 & & V6 \\
& & & V7 & V8 \\
& & & & & V9 \\
& & & & & & & V10 \\
& & & & & & & & V11 \\
& & & & & & & & & V12 \\
\end{align*}
\]

Solution:

The attributes that compose a relation’s primary key are underlined.
• A(V1,V2)
• B(V1,V3)
  – FOREIGN KEY (V1) REFERENCES A(V1)
• C(V1,V4)
  – FOREIGN KEY (V1) REFERENCES A(V1)
• D(V5,V6)
• E(V1,V8,V9)
  – FOREIGN KEY (V1) REFERENCES B(V1)
• F(V11,V12)
• F_{V13}(V11,V12,V13)
  – FOREIGN KEY (V11,V12) REFERENCES F(V11,V12)
• R_1(V1,V5,V7)
  – FOREIGN KEY (V1) REFERENCES C(V1)
  – FOREIGN KEY (V5) REFERENCES D(V5)
• R_3(V1,V8,V5,V11,V12)
  – FOREIGN KEY (V1,V8) REFERENCES E(V1,V8)
  – FOREIGN KEY (V5) REFERENCES D(V5)
  – FOREIGN KEY (V11,V12) REFERENCES F(V11,V12)

Note that for entity D we arbitrarily chose attribute V5 as primary key out of the two candidate keys \{\{V5\},\{V6\}\}.

3 Extending an ER Model

Consider the following ER model about vehicles and their owners.
Task:
For each of the following constraints decide if it can or cannot be represented in the ER model (using the notation taught in the lecture). For those that can be represented, adapt the ER model and clearly state how you changed the model. For the remaining constraints state why it is not possible to represent them.

1. A company employs at least one person.
   Yes, it can be represented by a total participation of entity Company in the employs relationship.

2. Every vehicle is either owned by a company or a person (but not both).
   Yes, it can be represented with the following extensions. We create an entity Owner from which Company and Person inherit (total, and disjoint inheritance). We add key-attribute OID to Owner that replaces keys CID and SSN (under the assumption that they have the same domains). We drop the two existing relationships “owns” and replace them with a single 1:N relationship “owns” between Owner and Vehicle.

3. A company owns at least ten vehicles.
   No, this cannot be represented (neatly) with the given notation, in which only three cardinality constraints are provided: one-to-one (1:1), one-to-many (1:N) and vice versa, and many-to-many (M:N).
   However, a possible but awkward workaround is as follows: Create ten entities Vehicle-1 to Vehicle-10. Make a total and disjoint inheritance from Vehicle to the ten new entities Vehicle-i. Create ten relationships as follows. For the ith relationship create a relationship between Company and Vehicle-i with cardinality constraint 1:1 and a participation constraint such that Company must own Vehicle-i. This ensures that a company owns exactly ten Vehicles. The existing (1:n) relationship between Company and Vehicle ensures that a Company owns 0 or more Vehicles. Together, they ensure that at least ten vehicles are owned by a Company.

4. Every vehicle is either a truck or a car (but not both).
   Yes, it can be represented by making the ISA inheritance total (since “every...”) and disjoint (since “but not both”).

5. A person can have multiple SSNs.
   Yes, we need to make SSN a non-key attribute, since key attributes cannot be multivalued. An entity needs at least one key and since a Person now inherits from Owner (with key OID), also Person is identified by an OID. If we had not created this inheritance, we could have added a new attribute, e.g., ID, and made that key.

6. Functional dependency: PlateNr → VID
   Yes, it can be represented by making PlateNr a candidate key. This implies that PlateNr must functionally determine all of Vehicle’s attributes, including VID.