Exercise 1: TDT4145 - Data Modelling, Databases and Database Management Systems

alexaoh, jimoskar

January 2021

Task 1

a)

Broadly speaking, a database is a collection of data that are related in some way. A more restricted description of what constitutes a database is that it i) represents some aspects of the real world (miniworld), ii) it is a logically coherent collection of data with some inherent meaning, and iii) it is designed, built and populated with data for a specific purpose.

A DBMS is a software system which enables users to create, maintain and interact with a database.

b)

1. Program-data independence means that the structure of data files is stored separately from the programs that can access the data. This makes it more efficient to e.g. alter the description of a specific type of record in the database, because you only need to alter the catalog in the DBMS, not any program that accesses the data. This is because the application programs are (mostly) immune to changes in the data structure.

2. Multi-user support could refer to a database system's ability to view the database/data in different ways. This allows the database to be used by a variety of users, whom may require different perspectives of the database. Multi-user support could also refer to the database system's ability to let multiple users use the database at the same time. This requires concurrency control systems and is a major advantage of the database approach compared to traditional file-processing software.

3. The self-describing nature of a DBMS refers to the fact that the database system contains a complete description of the database structure and constraints. This is called meta-data and is stored in the catalog of the DBMS.

The advantage with this approach is that the DBMS becomes general purpose, i.e. it does not need to be application- or program-specific. In this way the DBMS collects the meta-data in each application, which specifies how the database should work in that specific case, and it need not be constrained to work only with one database. As long as the necessary database definitions are stored in the catalog, the DBMS is general purpose and will work equally well for all types of database applications.

Task 2

a)

1. The difference between an entity and an entity type (or class) is that the entity type can be thought of as a blueprint for all entities of that type, while the entity itself is one specific object instantiated from that blueprint. Moreover, the entity class is a set of entities, which means that no two entities can be equal. This will be further discussed in item 3. below. The relationship between an entity type and an entity strongly resembles the relationship between a class and an object in OOP: the class is the template and an object is an instance of the class.

2. The difference between a relationship and a relationship type is, again, similar to the difference between an entity and an entity type: the relationship type defines a template for how to model a relationship between two or more entities, while a relationship is an instance of that relationship type. Phrased differently, a relationship type is the set of all relationships (often referred to as the relationship set) among entities from the given entity types in the relationship type.

3. All entities must have one or more key attributes because, as already mentioned, each entity type is a set of entities, which implies that each entity belonging to the same entity type must be different. Hence, each unique entity needs a unique identifier (exception: weak entity types. These can still be uniquely identified, but their unique identifier is collectively defined with other entity types). More practically, the users of the database are not able to work with the data without these identifiers, since the entities cannot be identified in an unambiguous way.

b)

Claim #	Answer	Justification
1	True	'Taco' is an entity class with the underlined key attribute 'TacoID'.
2	True	From the min-max pairs we can see that a given taco can have anywhere between 0 and n different sauces, cheeses, vegetables and meat.
3	False	The min-max pair $(1, n)$ on the 'Order'-side of the relationship class 'TacoOrder' implies that an order needs to contain at least 1 taco.
4	True	$n = 10^6$ is a possible choice in the above min-max pair, since the n signifies an arbitrary amount.
5	False	A specific order can only be connected to a single shop, as the $(1, 1)$ pair on the 'Order'-side of the relationship class 'PickedUpAt' shows.
6	True	$(0, n)$ min-max pair on the 'Customer'-side of 'CustomerOrder' implies that a customer can exist in the database with no orders.
7	False	'Weight' is an attribute of the relationship between 'Taco' and 'Vegetable'. This means that the entity itself cannot have this attribute. However, this question is a bit ambiguous, since each 'TacoVegetable'-relationship could have a weight attributed to it.
8	True	'Job Title' is an attribute of the relationship between 'Employee' and 'Shop'. Moreover, since the min-max pair on the 'Employee'-side of the relationship is $(1, n)$ it means that each employee can work in different shops with different job titles.
9	Maybe	Each order is delegated to one and only one employee, which is seen from the $(1, 1)$ min-max pair on the 'Order'-side of the relationship between 'Order' and 'Employee'. But we do not have enough information to deduce whether or not all the orders are delegated to the same employee. This is however a possibility, since no part of the information in the model denies this.
10	Maybe	We have no information regarding the data types of each of the attributes. Hence a NULL-value may be allowed for attributes that are not keys, but we do not know if this is the case. This is a possibility however, since 'name' is not a key in the 'Customer' entity type.

Task 3

a)

We use weak entity types when there is no natural key attribute that can identify the entities of that type. Rather, weak entity types have an identifying entity type, with which it has an identifying relationship. A weak entity type always has a total participation constraint with respect to the identifying relationship, and can not exist without the identifying entity type. In the given model, 'ScreenRoom' is a weak entity type with the owner type 'Cinema' and the identifying relationship 'ScreenAtCinema'. A weak entity type normally has a partial key attribute, which can uniquely identify a weak entity among entities with the same identifying entity. In the given model, 'ScreenRoomID' is a partial key of the 'ScreenRoom' entity type. This means that, in this example, it is assumed that no two screen rooms in the same cinema have the same id, which is why the id of the screen room is modelled as a partial key, since it uniquely identifies each of the weak entities that are related to their respective identifying entities.

b)

If the cardinality from 'ScreenRoom' to 'Cinema' changes to $(0, 1)$ the entity type 'ScreenRoom' cannot be modelled as a weak entity type, since it does not satisfy the total participation constraint that is imposed on weak entity types. In such a case each 'ScreenRoom' entity would be in need of a unique key. If the cardinality from 'ScreenRoom' to 'Cinema' changes to $(1, n)$ the entity type 'ScreenRoom' could still be modelled as a weak entity type, since the total participation constraint is satisfied. However, in this case, each 'ScreenRoom'-entity could be related to many different cinemas, which is possible in theory, but perhaps does not make much sense in the mini-world prescribed by the ER-model (which we know nothing about).

c)

We assume that each 'ScreenRoom' and 'Movie' can exist without having a movie-relationship, and that each 'ScreenRoom' can be associated with an arbitrary amount of movies. Figure 1 shows the new ER-diagram to accommodate for the new specifications.

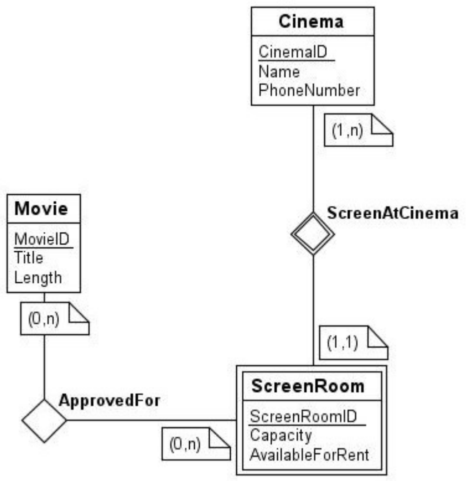

Figure 1: Expanded ER-diagram from task 3c), where the relation 'ApprovedFor' between the new entity type 'Movie' and 'ScreenRoom' is added. The underline of the key in the weak entity type signals that it is a partial key.

d)

The functional model is given in figure 2.

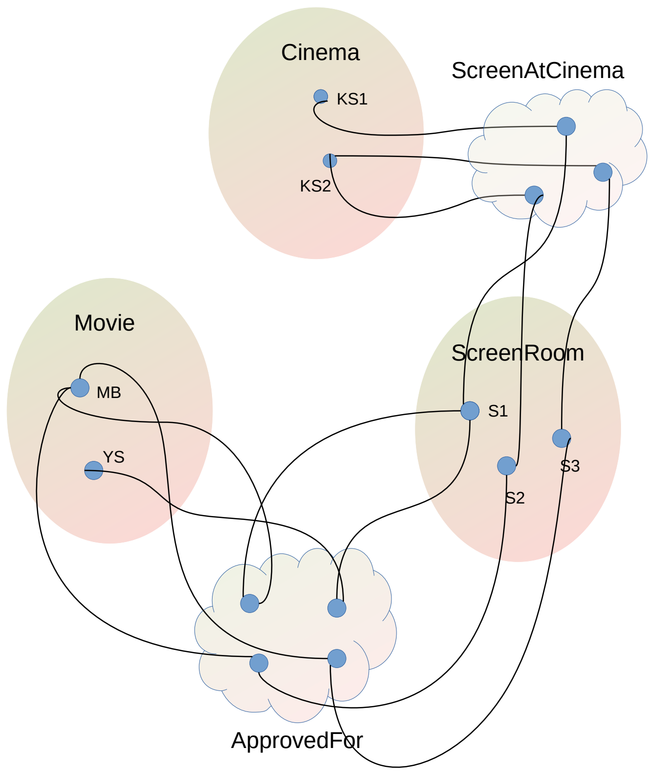

Figure 2: The described functional model from task 3d).

e)

Figure 3 shows the expanded model incorporating movie showtimes and associated tickets to the show. We assume that a ticket must have a relation to a 'MovieShowtime' for it to exist (one to one). 'Ticket' is a weak entity type because of this total participation requirement. Furthermore, the 'SeatNumber' cannot be a key in itself, an entity is only unique combined with the 'ScreenRoomID'. In addition, we assume that a 'MovieShowtime' can be added to one and only one 'ScreenRoom', and that 'ScreenRoom' can have an arbitrary amount of showtimes. 'MovieShowtime' is not a weak entity type, since it has a unique key in 'SerialNumberID'.

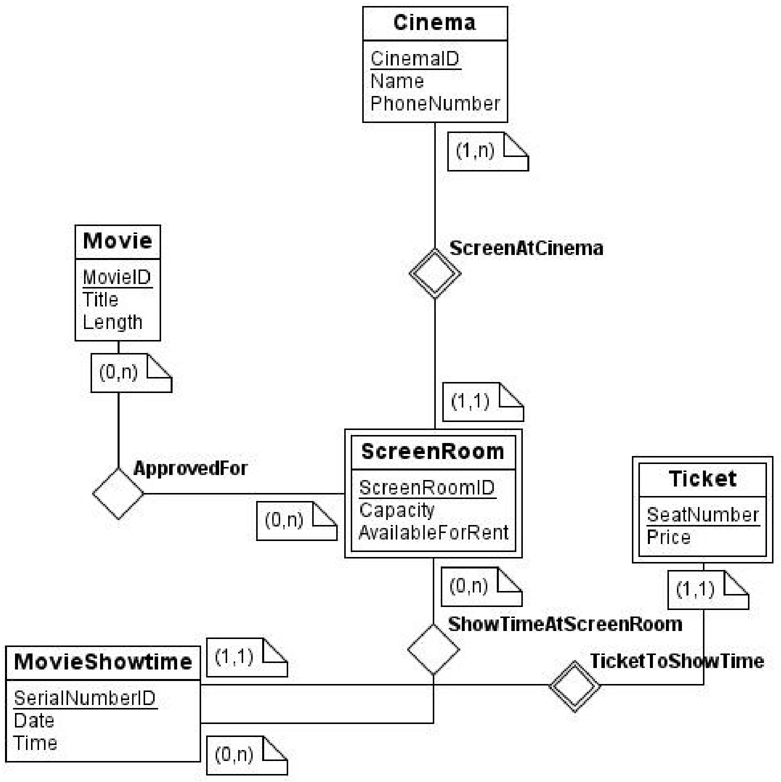

Figure 3: Expanded ER-diagram from task 3e). The underline of the keys in the weak entity types signal that they are partial keys.

Task 4

Below is a representation of the housing cooperative as an ER-model.

Important assumptions: We assume that a booth must be connected to a building, i.e. the Building entity type is the owner type of Booth. Building is also assumed to be the owner type of Apartment. A case is assumed to be related to an arbitrary amount of Person, Building and Apartment entities, and these types can also participate in an arbitrary number of cases. This results in the RelatedToCase relationship type which is of 4th degree. The Govern entity type must have one member, and we assume that if there is only one member, it must have Title leader.

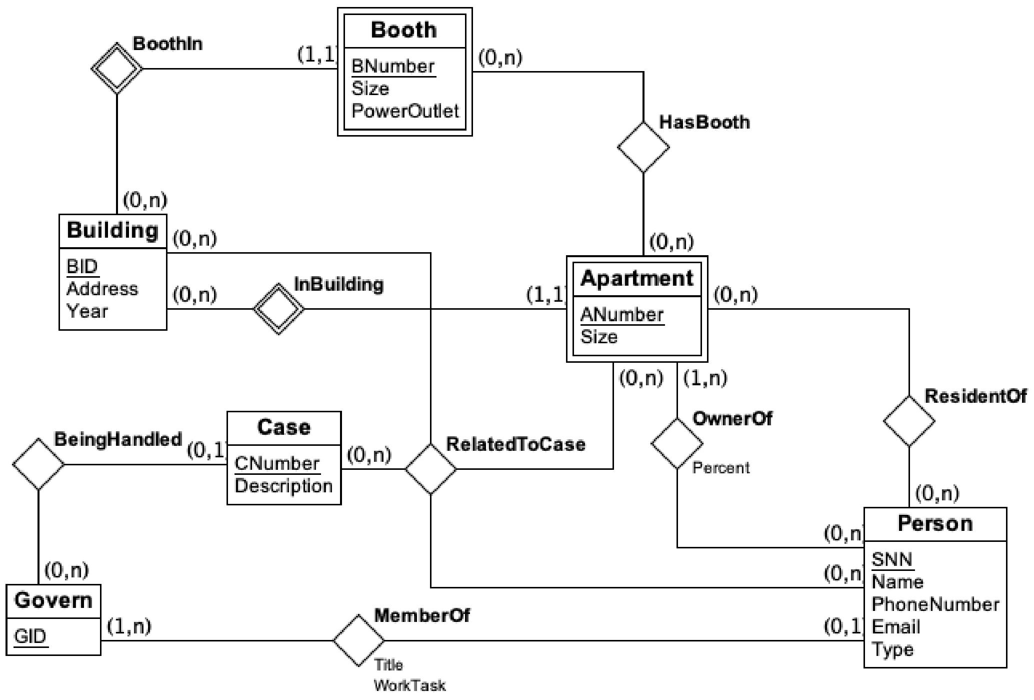

Figure 4: Representation of the housing cooperative as an ER-model.