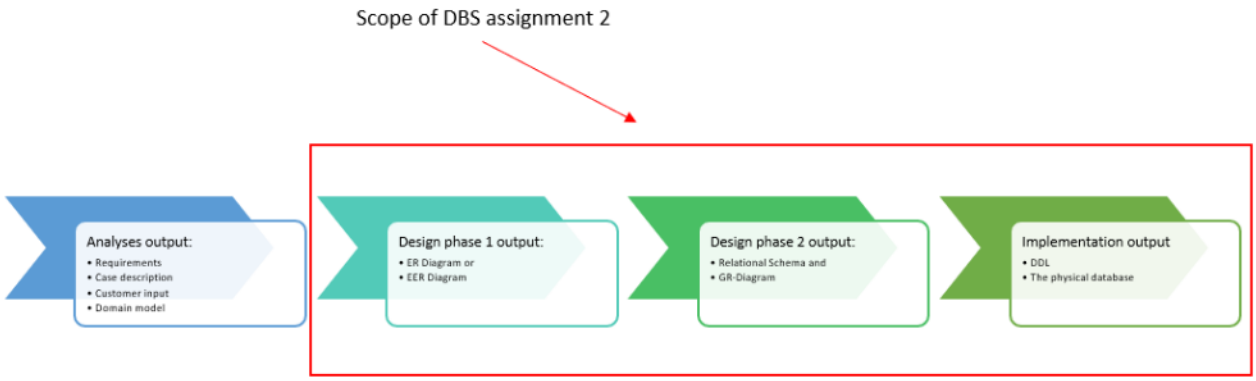
DBS Mandatory Hand-in 2

R. Brooks & T. Mortensen

Spring 2022

Introduction

The second of three mandatory hand-ins in DBS deals with ER Models, mapping this to the relational model[1] which is materialised in a relational schema and a GR-diagram, and then implementing the relational model in SQL using the data definition language (DDL). We could summarise this as the following process:



The first two assignments are taken from last year's exam. At the exam you will have exactly 2 hours to complete this part, i.e. you will have 2 hours to complete assignments 1+2 of this hand in. For practice, it may be a good idea if you try to do this as a solo practice exercise and then compare with your group members afterwards.

The assignments must be made in the groups that correspond to the groups which you defined in DBS Mandatory Hand in 1. Under each sub-assignment it is specifically written which material you should use, what the deliveries are, and how to upload. Make sure you read each assignment carefully.

Assignment 1: Exam 2021

All diagrams and similar should be handed in as a single pdf document. All diagrams must be UML, as taught in class.

All SQL code should be in one or more .sql files.

1a

DBU (Dansk Boldspil Union) is planning on digitising their data, and for that they need a database. However, they want to start small and then expand if the initial prototype is a success. They have given the below description of the information they want to start out with.

Description from the customer: We want to store data about football teams, players, their stadiums, and the matches played between the teams.

[1] We use the terms 'relational model' and 'logical model' interchangeably. The book prefers the term 'logical model'.

A football team has a unique name, and a ranking, and their shirt has multiple colours. The team has their home field at a specific stadium, identified by a name. We are interested in the address and number of seats in the stadium.

A match is played between two teams. We need the time, date and at which stadium a match was played. We also require how many goals the home team and the away team scored, and how many overtime minutes were added to the match. And we also want to easily see which team won the match (or if it was a draw).

A team consists of players. For these players we need first and last name, salary, and which number their shirt has. Depending on the position on the field of the player, we need a few extra pieces of information for statistical purposes: we need to keep track of how many goals the goal keeper saved, and the attacker scored, in a season. We have had some rough matches, and we need to see if there is a violent tendency amongst some of the players. To do that we need the number of red and yellow cards for the defenders.

Finally, we want to know which player scored goals in the matches, and at which match minute count the goal was scored.

Your task:
Create an EER diagram based on the above description.

1b

Now, take your EER diagram from 1a above and use the mapping rules to derive the relational schema. You must explicitly state the effect each step has on the mapping. If a step has no effect or is not relevant just state 'N/A'.

1c

Create the Global Relations Diagram based on the relational schema you made in 1b.

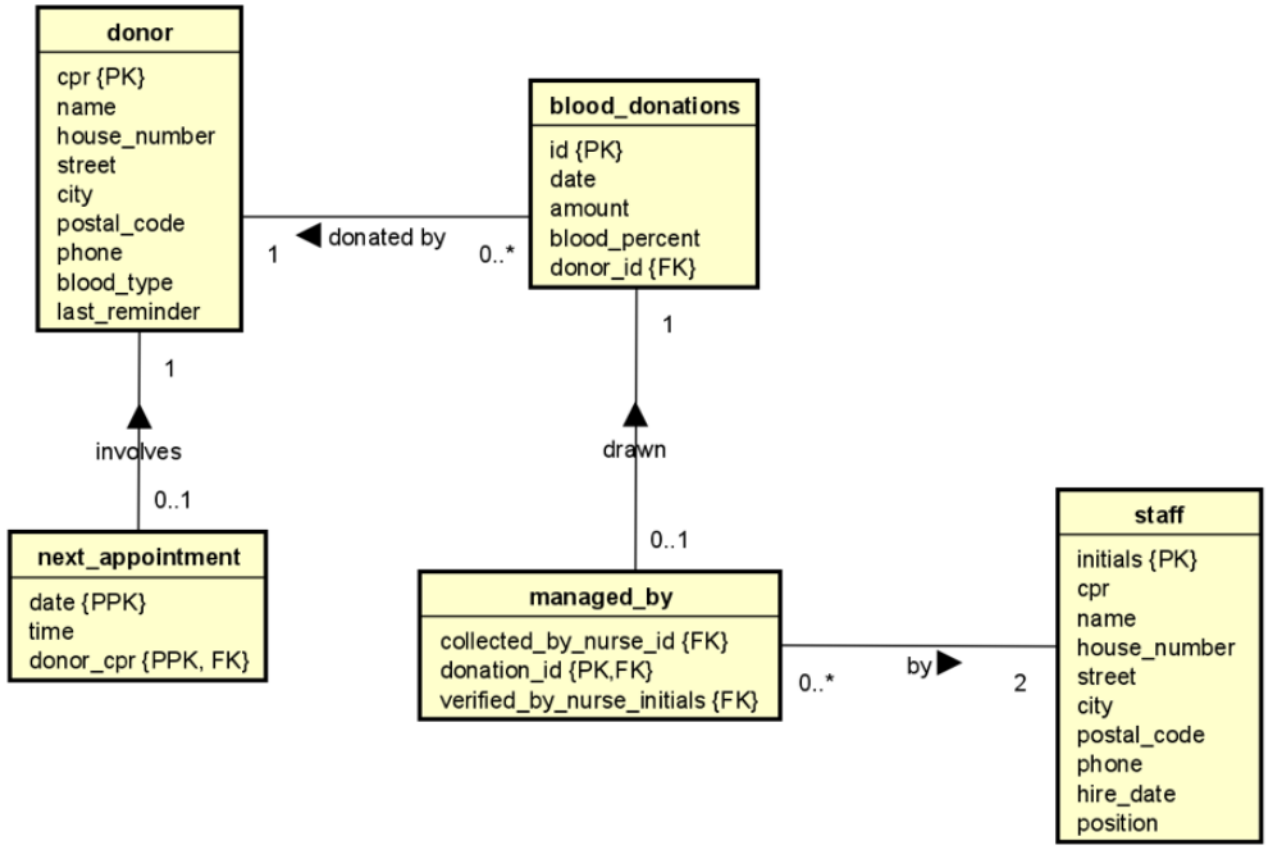
Assignment 2: Exam 2021

The blood bank of Horsens is in the process of digitising their data. They have started but they need your help to finish the task.

2a

Below is a Global Relations Diagram which is the product of their mapping of the database. Based on this diagram, and the comments below, implement the corresponding DDL code.

You must hand in the code in a .sql file.



Notes:

donor:

cpr: This is string of length 10.

name: The full name of the donor

blood_type: This is a max 3-character code, with the possible values of: O-, O+, B-, B+, A-, A+, AB-, AB+

last_reminder: This is the date when the latest reminder was sent to the donor to make an appointment.

blood_donations

id: this should be an auto incrementing integer.

amount: this is the number of ml of blood drawn, it is between 300 and 600 ml. No decimals are needed.

blood_percent: the blood percent must be between 8.0 and 11.0, and indicates the haemoglobin levels of the donated bag of blood.

managed_by
A blood donation involves two members of staff, one for drawing the blood, and another for verifying the information on the blood bag.

staff
position: There are 3 possible positions: nurse, biomedical assistant, intern.

2b

Create the DDL for inserting two rows of data into each of the following tables:

donor, blood_donations, staff, managed_by

Note, you need to come up with fictitious data in order to do the inserts.

You must hand in the code in a .sql file.

Assignment 3: Who are you - and which courses did you take?

For the third and final assignment we return to our 2nd semester database.

3a

Assume we made the 14-tuple table and movie table from "DBS Mandatory Hand-in 1" into an ER-diagram. This would result in an ER-diagram with just two entities! Clearly, this is not a good database design. In this sub-assignment we want you to assume that the 14-tuple table and the movie table are given to you as customer input instead of the usual description. Based on the data of those two tables, you must redesign your database. Start with an EER diagram of the 2nd semester Database, i.e. redesign all the information from the two tables into meaningful entities and output a new and better designed EER-diagram.

3b

Now, take your EER diagram from 3a above and use the mapping rules to derive the relational schema. You must explicitly state the effect each step has on the mapping. If a step has no effect or is not relevant just state 'N/A'.

3c

Create the Global Relations Diagram based on the relational schema you made in 3b.

Assignment 3d

We would like to augment the database with the following information which should ultimately lead to *at least* one new table.

Each class has had a number of courses during their first and second semester. We would like to keep track of these courses as well as when they were taken (Autumn 2021 or Spring 2022). Particularly, we want to keep track of the following information about each course:

- its name, e.g. 'Software Development with UML and Java', and its unique courseId (e.g. 'IT-SDJ1')
- one or more teachers that have a name and who are uniquely identified through a staffId (e.g. The staffId of 'Troels Mortensen' is 'TRMO')
- the number of ECTS points

- the experienced difficulty by the student
- whether there was an associated workshop or study café
- one or more student instructors (or perhaps none)
- when the course description was last updated

The students in this description reference the students in the 14-tuple table, and the student instructors are also students who have a name and a studentId but they are not present in the 14-tuple table. The experienced difficulty is a value from 1 to 5 where 5 means that the experienced difficulty of the course is very high and 1 means that the experienced difficulty of the course is very low (and 2, 3, and 4 means that the experienced difficulty is somewhere in between). We will not distinguish between workshops and study cafés and simply reference them both as 'workshops'.

Please note that the table should be compatible with information from other classes. For instance, we may have courses with the same id and the same teacher that are taught in different classes (e.g. RIB taught IT-DMA in both class X and Y in Autumn 2021). Your database must be able to distinguish between two such courses.

Your task:
Implement a table/tables that capture(s) the information above. The DDL code must include the create statements, including constraints, as well as any domain definition statements. The DDL must be structured such that any and all tables can be joined with the original 14-tuple and the movie tables. It is not necessary to explicitly state the design process for this part of the database, although to optimise the database it may be a good idea to consider the design process which you went through in assignment 1 as well as in 3a-3c. It is important that you minimise data redundancy as much as possible. In the final sub-assignment below you have to include the information about the courses in the GR-model of the database which you made earlier.

You must hand in the code in a .sql file.

3e

Create insert statements for the table/tables created in sub-assignment 3d. In other words, fill in the correct data just as you did in the very first case about the 2nd semester 14-tuple table and place the data directly into the insert statements. Each student must make an assessment of each course such that the student is able to give the courses a difficulty rating as described in 3d, i.e. a value between 1 and 5. The table(s)from 3d must thus be populated and we must be able to query them and receive a sound result.

All the information you need to populate your database is readily available, e.g. information about when the course description was last updated can be found in the course description.

You must hand in the code in a .sql file.

3f

Augment the GR-diagram you made in 3c such that it now includes the relations relating to courses from 3d. That means that the GR-diagram must contain relations that capture the information contained in the 14-tuple table, the movie table, as well as any table you created in 3d.