Seminar 2 - Logical and Physical Models

Data Storage Paradigms, IV1351

Kevin O'Flaherty, kevinof@kth.se

December 9, 2022

1 Introduction

For the second seminar we built on or earlier work with the CM for the Soundgood musical school. This time, we were expected to produce a Logic/Physical model, using UML or other crow foot notation. We were then to take this model and implement it by creating a corresponding database. These were to be exact mirrors, so any change discovered to be necessary in the implementation were expected to be fixed in the model as well.

For a shot at a higher credit, where inheritance between entities modeled in the conceptual model there should be an exploration and comparison of how their respective tables were implemented in this task. Also, the Discussion chapter in this report should reflect relevance and depth in relation to the issues addressed.

For this task I started off by refining the previous CM post seminar 1 working with Vilhelmina Andersson, Patricia Lagerhult and Johan Seller-Fredlund to go over relations between entities. I also turned to them whenever getting stuck on a thorny issue.

2 Literature Study

In advance of starting the task I watched the videos provided in the lecture on Logical and Physical Models. After watching the prerecorded lectures on normalization and finding them hard to follow I turned to a video on the topic, found at `https://www.youtube.com/watch?v=GFQaEYEc8_8`. I found this video more practically oriented in relation to the normal forms, which suited the task at hand better. It also gave me a more solid ground to stand on, when returning to the lectures on the subject as well as the written material in Fundamentals of Database Systems.

From the same textbook I read about how to translate the CM model's defined inheritance between certain entities to the logical-physical model. More on that in the Discussion chapter later.

I also read the course material pdfs tips-and-tricks-task2 (specifically to tackle the self-relation between student and sibling) and assessment-criteria-seminar2.

3 Method

Since I used Astah to write the CM in the previous assignment I decided to continue with this program for the logo-physical model. I more or less followed the eleven steps outlined in the online lectures.

This led to a diagram with tables for each entity and each column with zero-to-many cardinality or higher, as well as cross-reference tables for many-to-many relations between entities. These tables were filled with column for each attribute, before moving on to specify their domains and consider the column constraints. (More on the types in the Discussion chapter of this document). All strong entities, as well as weak entities with intrinsic meaning were assigned surrogate keys to identify them. These primary keys were given the type INT with the classification GENERATED ALWAYS AS IDENTITY to later be read in the database as unique identifier generated as data was added. Other weak entities were generally given the these keys as foreign keys all the while considering deletion constraints. This was especially true for the cross reference tables where the FK:s constituted their PK.

The next step was to verify that the model represented a sufficiently normalized database. To this end I followed the lecture material, but also the youtube video referenced in the Literature Study.

Following this I read through the business description of the school to make sure all necessary operations could be performed, and arrived at a foundational logo-physical model. I used Astah's tool to export the model as an SQL-file, but trying to load it directly as a database produced several errors. These are discussed later, but required some backwards and forwards adjusting between the script and the model to bring them inline with each other. I also added to the script the functionality mentioned in the diagram's red notes (ON DELETE CASCADE and UNIQUE constraints), which of course weren't translated automatically to the script.

Once the database didn't throw errors at creation the next thing to do was fill it with data. Sadly I ran low on time, and so while I did insert some data I'm not happy to call it a filled database.

4 Result

The following diagram can be found in its most recent form together with its corresponding SQL script at the following github repository:

`https://github.com/kaytaffer/soundgoodDB`

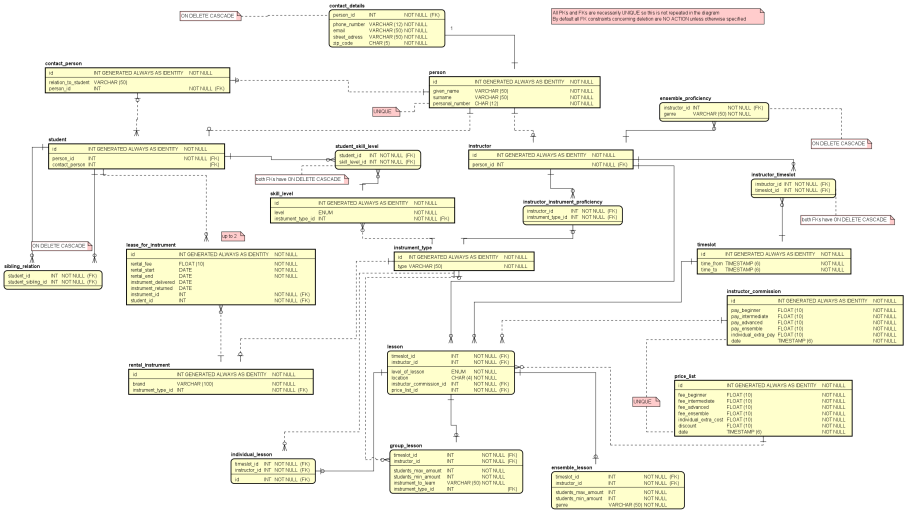

Figure 4.1: The model for the Soundgood Music School DB

As the diagram details may be hard to read in this document I recommend downloading the image titled *Log − physmodel.png* located on github. The model closely mirrors the conceptual model it's based on, but has more tables than the CM had entities. This is certainly true if one considers the *instructor*, who extends person and has many-to-many relations expressed through lookup tables as well as started out with attributes with one-to-many cardinality.

The script that sets up the database is found in *seminar2dbscript.sql* and the script that inserts data is found in *insert.sql*, however at present I wouldn't say it's very sophisticated.

5 Discussion

As I was creating tables corresponding to the entities in the CM I ran into a similar problem as Lindbäck tackles in the lecture concerning the entity *address*. In my case it was contact details that had 0-1 relation with both the *contactperson* and *person* entities, suggesting that it was a strong entity in relation to the person to whom the contact details related. This seemed backward to me and I first considered going the slightly denormalized route that Lindbäck implemented in the lecture video. While solving my issue, this would have meant deleting the contact details table, and create duplicate data entries, risking a format change of the way contact details are stored to have to update it in multiple places.

I finally decided on doing something I perhaps should have done earlier. In the creation of the CM i opted to not let *contactperson* inherit *person* on the ground that the contact

person didn't need the attribute personal number. Rethinking this, I decided that since I don't have a client preference to state otherwise, the most elegant solution was to make a $contactperson$ a $person$, leaving $contact_details$ to all $person$s in its own table.

Another issue I ran into while considering strong entities and their weak counterpart was the relationships surrounding a $lease_for_instrument$. It does not make sense to have a $lease_for_instrument$ without a leasing *student* or a $rental_instrument$, so it is clearly not a meaningful entity in its own right. However, it is conceivable that a single student leases the same instrument multiple times, so using the respective strong relationships foreign keys as the lease's primary key could result in multiple rows with the same primary key, which would be horrible form. The instrument lease therefore got a surrogate key to solve this issue.

In contrast I found that a *lesson* can be uniquely identified with a set of foreign keys since no *one instructor* will be able to lead multiple lessons on a specific $time_slot$.

Moving on to the issues concerning Astah and SQL, I found that defining INT GENERATED ALWAYS AS IDENTITY as a type worked fine for primary keys using Astah's SQL export, but since the foreign keys this generated for dependent entities 'inherited' the type in the script, this threw lots of errors. The only thing to do was go back to the diagram and redefine the FKs to regular INT. My other self defined type, ENUM, in the case of the skill level for students, also generated errors. In this case, the solution was to manually add the following line to the script:

CREATE TYPE SKILL AS ENUM ('beginner', 'intermediate', 'advanced');

I then could set the level as a SKILL in the database.

I considered defining instrument type as an ENUM since an argument can be made that they should be considered discrete and limited values. However the sheer amount of possible instrument types makes it impractical to list them all. Also since new kinds of instruments are likely to develop as the state of the art does, it seemed prudent to leave it as a VARCHAR.

This final paragraph is a stub, representing the fact that I due to time constraint didn't discuss the use of inheritance in this document as planned, as well as my intention to finish this in seminar 5.