SoundGood Music School Rapport

viktor björkén

November 2022

Introduction

In this part of the project our goal is to make our conceptual model into a "Logical/Physical model". We are then going to use this model and with the help of Posgres make a SQL script that constructs this database. We are also going to add data to the database with the help of generated data to make sure everyhing works correctly. Me and my partner Mattias Sandberg are working closely together during this project to try and help eachother to find the best solution.

Method

During this part of the project our first goal was as mentioned above to create a new model. Just like our conceptual model we used the diagram editor *astah* to make our model as it was suggested by our course literature.

When we created our model we followed the online lectures about logical and physical models. The luctures fetured eleven steps on how to create such desired model where we tried to follow them all. Some of the main steps where as follows.

- Tables and Single/Multi Attributes
- Column Types and Constraints
- Primary Keys
- Different Relations
- Normalization
- Verifying

We start by creating a table for each entity for our conceptual model from the previous assignment. One other thing that differs from the previous model is that we now create individual tables for multi valued attributes instead of having them as attributes under one entity. One thing we have to keep in mind

is that now our naming convention has changed. Within this model we instead of using upper and lower case letters we instead only use lowercase underscores.

The next crucial step is to set a specific datatype to each column. This could for example be int, varchar or char. Depending on what type of column we are dealing with different data types can be more suitable than others. While we are changing our types we can also set up our constraints, constraints are used when we only want a set number of entries in one column for instance will our personal number always be 12 numbers. To do this we will be able to use our different types and set boundaries on how many characters you can enter, either a maximum value or a strict value.

Thereafter we set our strong entities as primary keys, where a strong entity is an entity that can exits on its own without being reliant on other entities. When we perform this step we also create a surrogate key, this is a unique entity essentially only used for this purpose called for for instance "id". We do this to keep the primary key from changing which could be a possibility if we had entities with double usages.

In the next step we will add our different relations between the different entities. Depending on whether or not an we conciser an entity strong or weak we set foreign keys or primary keys. This is where we set either one-to-one, one-to-many or many-to-many.

When it comes to normalization our new model following the steps from the recommended lectures our model should follow the at least up to the third level of normalization which is sufficient. We do a double check to make sure that all steps are for filled.

The last step is to also double check all the operations from the project description are adequate.

Result

The model we made according to the steps explained above can me seen in "figure 1". We started by recreating our tables and single/multi attributes from our conceptual model. This was a bit hideous but worked out smoothly. Because of this we got a few more entities that we needed to handle. For example we now needed a new entity for the phone and phone number instead of having it as an attribute within a class like a the previous model.

This new logical/physical will now check of more boxes needed to be able to fully develop our database. Some of the changes that we did where to add primary and foreign keys this was something that we where a bit unsure of but we decided to for instance use a primary key "student_id" from "parent". This was used so that we easly can match a student with its parent.

Down below here you will be able to get directed to our github repository where our script for implementing and inserting data into the database will be stored.

Github link

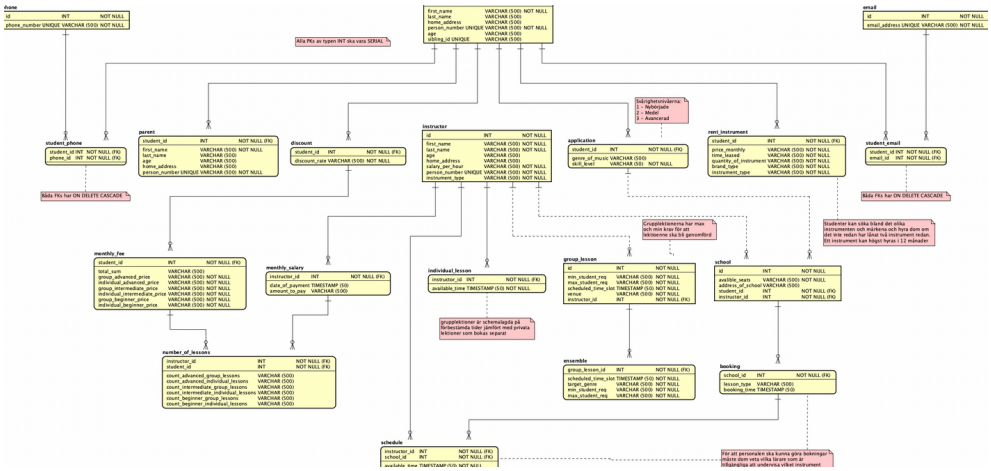

Figure 1: Logic/Physical Model

Discussion

When creating this model we tried our best to follow the lectures and course material that was given to us. Beyond the lectures about logical and physical models, we also noted the lecture about normalization. The lecture about normalization was not coherently connected to the task we worked on to such a high degree but rather a good lesson on how and why normalization is used. Instead we opted to using the steps on how to construct this new model excessively. We didn't stumble upon any major issues follow the steps and are satisfied with how the model turned out. Where we did in fact run into errors where when we during the next step tried to convert the model using SQL scripts. This was something non of us had previously worked with, due to this fact we had to opt to using other sources to try and help us solve this. We found that we could use a GUI like *pgAdmin* to create our database without actually writing any code ourselves. We realized that if time was available it would be favorably to write it manually to better understand what was happening, which is was we settled on doing.

Like I previously mentioned we tried to follow all the steps accordingly, some aspects that can be disused are crow´s foot notations. Seeing as we formerly made our model using a "class diagram" we needed to learn and correct the model to use the correct notation. Another aspect that we could discuss is if all tables are relevant. I think we did i decent job at only leaving and what is necessary and removing excess tables, perhaps some information could be calculated without the need of its own table/entity however to make it more clear and not accidental leave anything out.