Project Report - Task 2: Logical and Physical Model

Data Storage Paradigms, IV1351

November 19, 2024

Project members:
Nils Ekenberg, nilseke@kth.se
Samuel Engbom, sengbom@kth.se
Peter Li, pli2@kth.se

Declaration:

By submitting this assignment, it is hereby declared that all group members listed above have contributed to the solution. It is also declared that all project members fully understand all parts of the final solution and can explain it upon request.

It is furthermore declared that the solution below is a contribution by the project members only, and specifically that no part of the solution has been copied from any other source (except for lecture slides at the course IV1351), no part of the solution has been provided by someone not listed as a project member above, and no part of the solution has been generated by a system.

1 Introduction

Our task is to use the Conceptual Model (CM) of the Soundgood Music School from task 1 and create a logical model with enough physical aspects to create an actual database. As well as building a script that creates the database and a script that populates that database with dummy data.

2 Literature Study

In the given lectures as well as the initial part of chapter 9 in the coursebook *Fundamentals of Database Systems* we learned how to create a database from a CM. When going from a CM to a logical/physical model the details are expanded sufficiently such that you can construct a database based on the model. The logical aspect of the model deals more with tables, attributes, and relationships while a physical model expands upon the logical model to adapt it for database management, including implementation details like data type and constraints.

For this, the videos from Leif Lindbäck details how to make this kind of hybrid model with the Normalization lecture recorded on canvas explains more in depth about the concepts used.

3 Method

The logical/physical model was created using the Astah software tool. The process was based upon the recorded lecture videos where we leveraged our existing CM to create relations and attributes. The first step in converting the CM to the logical/physical model model was to add the existing entities as tables and change the naming conventions of these. The next step was to add all the single-valued attributes followed by creating new tables for each column of a higher cardinality.

After adding in constraints and types we assigned keys and relationships. We had relationships containing multiple records associated with another table so to model this we needed junction tables to break them into two one-to-many relationships, normalizing multi-valued attributes in to separate tables.

To normalize the model we followed the steps of 1-3NF (Normalization Forms). Made sure that each column contained atomic values (1NF), ensured that attributes had a functional dependency to the primary key (2NF), and made sure that there were no transitive dependencies (3NF).

The database script was written by hand in Visual Studio with the aid of using pgAdmin 4's GUI to easily spot mistakes.

4 Result

By using the methods as mentioned in previous section, we were able to create a logical/physical model for Soundgood Music School at this stage. The model is shown in Figure 1 below.

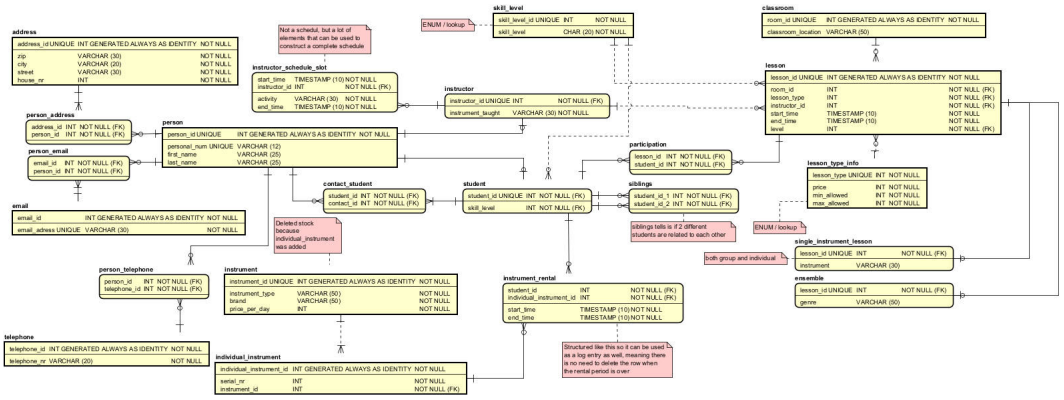

Figure 1: The logical/physical model

The database was built in Visual Studio. We generated the database files and inserted some sample data for the database. Link to the GitHub repository containing the creation and insertion files: `https://github.com/yyIntsuki/data-storage-paradigms/tree/main/task-2-sem-5`.

5 Discussion

When creating the logical/physical model, upon further discussions, we decided to remove some of the entities that existed in the CM as we realize they were redundant. The methods of normalization was of great help for reducing redundancies.

In the logical/physical model, we used cross-referencing entities that were necessary to describe many-to-many relations between entities, such as between `student` and `lesson`, a cross-referencing entity `participation` was created. Another case is between `person` and `student`, where we created a `contact_student` entity.

There is an entity to describe the relation between a student's siblings using two relations both from the `student` entity to an entity `sibling`.

For the formats, we followed a consistent `snake_case` naming convention. The IE crow foot notation was strictly followed.

Even though transferring the logical/physical model seemed like an easy task, in reality it was more complicated as it involved also learning how to properly input queries into pgAdmin and to export the database into SQL files. There were also various problems that the group members stumbled on, such as unable to backup the database in pgAdmin for export such that other members can import to their pgAdmin and run locally.