Seminar 2

Data Storage Paradigms, IV1351

Ayda Ghalkhanbaz
aydag@kth.se

November 24, 2023

Contents

Introduction

In the second task of the project, a logical and physical model based on the conceptual model from the previous has been designed. The logical model is then used to create a database in Postgres. The entire process of designing the logical model and creating the database is covered in this report. Additionally, the results of this task will be discussed in later sections.

Literature Study

The study material used to complete this task was the recorded films about the logical and physical model along with the lecture on normalisation. These materials provided the foundation in order to understand and accomplish the task.

Method

The method employed for this task includes eleven steps, which is mentioned and explained in the videos on logical and physical models. The process started with deleting the relations from the conceptual model, which was designed in the previous task, leaving only the entities. After that, every entity was checked to see whether all attributes are atomic (1NF). This step led to introducing some new entities. The next step was to define a type for each attribute.

Following this, the primary keys for each entity were identified, new relations were drawn between tables which resulted in defining foreign keys for several entities. However, achieving 2NF also determines the FKs constraints, i.e. how the updating and deletion of the FKs should be handled.

The last step was to achieve 3NF. This step might not be completely fulfilled since there is a data transitivity within the address attributes of the person entity, which, according to the lecturer, is acceptable.

After reviewing the model, a SQL script was extracted from ASTAH. Although the script needed some modifications, it provided a good foundation of the database script. It is noteworthy that during the modification of the SQL script, multiple flaws in the model were discovered, which led to further improvements in the logical model.

Result

The logical model designed for this task is illustrated in figure 1. The main differences between this model and the one designed in the previous task is that multivalued attributes now have their own entities with unique IDs. This results in less redundancy of attributes in tables since any table that wants to use that attribute can have a FK to it. For instance, `skill_level` only includes 3 different levels, and instead of specifying the level everywhere it's needed, one can refer to the skill level's ID.

Another notable difference is that there is now a `sibling` entity that stores student's ID and and their sibling's student ID. Additionally, two many-to-many relations have been defined. One is between students and lessons, and the other one is between instructors and instruments. Students may want to book several lessons and a lesson might be booked by several students (excluding individual lessons). Since this is a many-to-many relationship, the entity `lesson_booking` was introduced to handle the bookings and to save the record of which student books which lesson. There is a similar explanation for the instructor-instrument relation. An instructor may know several instruments and an instrument may be taught by several instructors. Hence, the `known_instruments` was introduced to handle this relationship.

As observed in figure 1, the `lease_policy` entity contains the conditions for instrument renting, that is, a student can only rent up to 2 instruments for a maximum period of 12 months. These conditions are recorded as a description for each contract. As another part of the higher grade task, two additional attributes were added into the `price_scheme` entity, i.e. `valid_from` and `valid_to`. These attributes set a duration for the prices, indicating that the specified price is valid for the mentioned period. Since the school might want to change their price list, these attributes allow the school to keep a record of their price history.

Also, the SQL script and a primary set of generated data to be inserted to the database can be found on GitHub.

Discussion

According to the lectures on the logical model, the naming conventions for both entities and attributes are followed. That is, all the names should be in lowercase and separated with underline. As we can see in figure 1, the crow's-foot notation of the ER diagrams is used correctly as well.

Regarding normalisation, as mentioned in earlier sections, the 1NF and 2NF rules have been applied to the model. However, the 3NF rule might not be fully accomplished in this model. For instance, the city, zip, and street (address attributes) in person entity are considered transitive data which doesn't completely fulfil the requirements for 3NF. Although having address attributes might be acceptable according to the lectures, it might be better to have a separate entity for the address, as there is a chance that several persons live at the same address. Beside this flaw, it is believed that there is no other place in the model that doesn't follow the 3NF rule.

All primary keys within each entity are unique within the same table, thanks to the SERIAL type automatically generated in Postgres, has been applied to them. Moreover, the constants such as skill levels, lesson types, rooms, genres, lease conditions etc. are defined in a separate entity with unique IDs as well.

Storing all data in the database has both advantages and disadvantages. It is, of course, easier and faster to maintain data, while simultaneously avoiding data duplications, due to the centralised nature of data in databases. It is also a safe way to store sensitive data, since databases provide secure access authentications. One other advantage can be the convenience of retrieving data in order to analyse them.

However, storing all data in the database is not all good and is followed by disadvantages. For instance, the application may require constant access to certain information, but the absence of data storage within applications can reduce the performance flexibility and speed. Therefore, it says that the application is highly dependent on the database. One other disadvantage might be the potential performance overhead, that is, in cases where reading and writing huge amounts of data occurs continuously which might require better network latency and query optimisations.

Recording multiple versions of data also follows with some advantages. For example, it enables a convenient way of historical analysis where the changes of the data over time can be traced easily. It also comes with this benefit of the data recovery, which is vital in case of errors and data corruptions. On the other hand its draw-backs are undeniable, such as storage overhead, higher complexity, weaker performance etc.

The choice of storing all data and recording multiple versions of data is a trade-off between security, flexibility and data consistency, which is ,of course, also influenced by the requirements and other circumstances.

Attachments

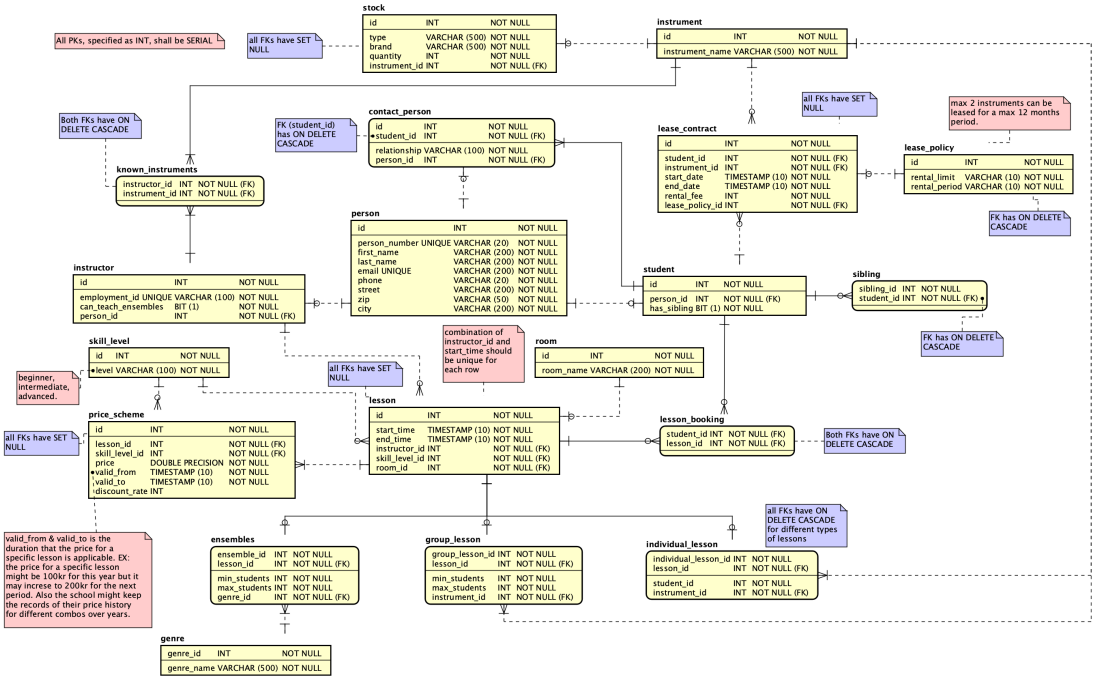

Figure 1: Logical Model designed for the SoundGood music school project