Project Report - Seminar 2

Data Storage Paradigms, IV1351

2024-11-14

Project members:
Simon Lieb Fredriksson, simonlf@ug.kth.se
Sebastian Taavo Ek, sebte@ug.kth.se
Adalat Adiljan, adalat@ug.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

This seminar report involves the solution for task 2, which includes the solutions for both the mandatory and the higher grade part. The general purpose of this seminar was to normalize the conceptual model and then transition it into a logical-physical model amalgam. One important note for the higher grade task was that the end product needed to account for the fact that pricing schemes both for renting an instrument and attending a class at our fictional Soundgood music school need to be alterable with time, without effecting earlier financial records. The pricing schemes also needed to reflect whether or not that particular price model was still active or whether it had expired.

Once the logical-physical model had been produced, the next step was to export it as an SQL file with a script to populate a database with the tables from the model. The export was largely functional, but needed to be amended only somewhat where the

code was faulty or incomplete. The final step of the task was to create an SQL script to generate dummy data in all the tables to make it possible to browse the relations in software such as PostgreSQL (psql terminal) and verify that everything was in order.

2 Literature Study

Key to our understanding of how to proceed were both the pre-recorded video lectures given by Leif Lindbäck, found on the course canvas page, as well as the course book "Fundamentals of Database Systems" by Ramez Elmasri and Shamkant B. Navathe. We will discuss our key findings from the study of both.

2.1 Leif Lindbäcks lectures

These lectures were our primary source of information for the assignment task. As Leif explains, going from a conceptual model to a logical-physical one meant that we were finally not just modeling our universe of discourse, but rather an actual database [2]. This meant that we could start to consider more database-oriented questions such as how best to optimize the design for search queries and developer readability.

Leif explains that the database design process would normally have you move from a conceptual model to a logical one, and only once the logical model has been produced do you take it one step further to a physical one. In our case, as the lecture describes, we group together these latter two steps into one in order to make the assignment more approachable for the student. Regardless of this shortcut, it would still be beneficial for our learning to briefly touch on its parts to understand this amalgam of design steps.

2.2 Conceptual to logical

The process of transitioning from a conceptual model to a logical one can be summarized by determining appropriate relations (tables) to use, followed by ensuring that the model is appropriately normalized [2]. Normalized tables ensure that the attributes of the tables are well-structured and free of partial, transitive or other dependencies, which could otherwise lead to inconsistencies or inefficiencies in the design. [1, page 475] As we understand it, in practice, this will lead to bigger relations being broken down into smaller tables where all attributes have a minimum amount of functional dependencies to attributes other than the primary key. Importantly, the decomposition of a table into a set of smaller tables containing the original subset of table's data must be a "lossless" transformation. [1, page 476]

After the logical model is normalized, the next step would be to make sure that all the specified operations from the project description can be done with the data in the new model, and only lastly add attribute types if necessary for clarification purposes.

2.3 Logical to physical

The physical model tells us **how** the data is stored, in addition to finally being adapted to a specific database management system like PostgreSQL. The physical model also specifies storage details such as column types, keys, indices and views. In the task for this report, we would end up glossing over the indices and views part of the process. The goal for the physical model is to produce a result that can be used right away in our actual system. [2].

2.4 The logical-physical amalgam

For this assignment and report, Leif mentions that we will use a different method that ignores security, indices and views, but still looks at the types of columns and primary keys that would be in in the physical model.

Leif outlines a set of steps to follow in order to appropriately transition our model from a conceptual one to a logical one with physical qualities. Without listing them in full, they largely resemble the steps present in chapter 9 of the course book [1, page 289] and involve creating a table for all entities present in the conceptual model, while only preserving atomic attributes with cardinalities between 0 and 1. Multi-variable or complex attributes are then represented as separate tables instead, and the developer moves through all tables in the model while assigning primary keys (using surrogate keys when appropriate) as they go, and foreign keys to represent a 1:1 or 1:N relationship between any two tables. N:N relationships are extracted and made into a separate table in which each column is a foreign key from the entities that had previously been connected through a relationship. The last step was to define any column constraints or foreign key constraints so that the database could appropriately handle anomalies or restrictions on the data such as allowing an attribute to be NULL or not. [4]

One final, interesting difference between the recommendations from Leif's lectures and the course book description is that Leif advises us to represent a 1:N or 1:1 relationship between entities in the conceptual model as a similar relationship in the logical model with the foreign key of either table in the other. An example for this would in our case be the "Instrument" table which has a "Student" foreign key in it, representing the student who is currently renting this individual instrument tuple. In the conceptual model, this was a 1:N relationship, and the logical-physical model solidified it through inserting a foreign key into the "Instrument" table. The course book, however, seems to be saying that cross-reference tables should be used (also known as look-up tables) even for these relations in some cases. The book mentions that this could be useful for cases where few tuples in one of the involved entities participate in the relationship, to avoid excessive NULL values in the foreign key column.[1, page 294] We interpret this to mean that if the the school's inventory of instruments were in a state of mostly not being rented out, it would be preferable to represent the relationship between students and instruments as a lookup-table instead to avoid excessive NULL values.

3 Method

As mentioned in the literature study section, we followed Leif Lindbäck's eight steps to convert our conceptual model into a logical one with some physical attributes. The diagrams were created and edited using Astah Professional under the KTH license. Subsequently, the SQL code was exported through Astah and the tables were then populated with dummy data using a second script, written in VSC.

Building on the previous section, we began by creating a table for each entity in the conceptual model, including attributes with a cardinality of 0 and 1. All other attributes were given their own separate tables, as were all relations with a cardinality of N:N. All entity relationships were represented through similar crowfeet as in the previous design, with a foreign key on the receiving side of the relationship referencing the primary key of the entity on the opposite end. The cross-reference tables for all N:N relationships were constrained such that each column represents a foreign key from the entities involved in the relationship in the conceptual model.

Lastly, we defined attribute constraints like uniqueness and "NOT NULL", as well as foreign key constraints. These are represented in the diagram using notes attached to the relationships that import the foreign keys. We tried our best to let all primary keys be surrogate keys when possible - following Leif Lindbäck's suggestion in the recorded lectures.

4 Result

4.1 Changes from the conceptual model

The logical model we produced managed to largely resemble the skeleton of the original conceptual model, but includes several additions and noteable differences. See the figure on page six for the model in full. First of all, we decided not to adopt the inheritance strategy between "Person" and "Student" or "Instructor". Instead, we chose to maintain "Student" and "Instructor" as separate tables, with no overarching definitions unifying them. As Leif mentioned in his lectures for the previous assignment, inheritance can sometimes introduce unnecessary complexity, and in this case we found no advantage to including a "Person" relation in the model. Maybe if someone wanted to query all the people associated with the school, students and instructors alike? But even then we believe that this can be effectively achieved using a join predicate.

Another change to the conceptual model was to make sure it tracks the cost of renting an instrument. Thus, we created a pricing scheme entity similar to the one for lessons, which can be edited at any point by school administration to update the instrument rental prices at any given time. We also added the ability to track individual pricing schemes, with specific expiration dates and start times so that administration can see what pricing schemes are still valid and for which transactions they were valid in the past. Finally, the biggest difference was the complete removal of the entities InstructorPayment and StudentPayment upon Leif's recommendation, since these entities were comprised entirely of derived data which we had missed.

The final changes made to the diagram included extracting the contact person's name and phone number from the "student" entity/table into its own separate table, with a foreign key `student_id` used as its own primary key. Furthermore, we made a new table for ensembles, separate from lessons. We took this decision because otherwise every tuple in the lesson table that was not an ensemble would need to hold a "NULL" genre column - which did not seem like an elegant solution at all. Since the "ensemble" entity was extracted and given a life of its own, it also needed the same helping entities as "lesson", and was therefore given the lookup tables `student_ensemble` and `ensemble_fee`. In the end, the logical model was quite transformed from the conceptual one, and it caused for us to have to revisit the conceptual model and update it with the changes we made to this one.

4.2 The model and FK constraints

The model follows the instructions laid out in the Method section, and tries to thoughtfully apply FK constraints in such a way that we only cascade delete a tuple in a table if the data point that the foreign key points to is essential to defining that tuple. For example, it makes no sense to keep an entry in the look-up table between student and address if either the student or address was deleted - allowing cascade deletes for those foreign keys. On the other hand, for all financial records we might want to save the history of the transaction even if a student is deleted from the database, even if that would obviously make the data slightly more incomplete to its context. We briefly thought about adding two additional entities to represent the transaction history and "complete" payments for lessons and rentals, such that we could save individual attributes of other relations like a student's full name in relation to a purchase even if that student got deleted. Ultimately, we decided against this since the tables would present derived data. However, this decision means that bits of the transactional record could be lost if students, lessons or even pricing schemes are deleted or lost. In a well-designed database, we imagine that you might treat this data as critical, backing it up and decoupling it from its original "references" (using this word loosely) and duplicating it in new separate tables, even if this means a certain degree of redundancy.

As for the SQL script to create the tables for the database as well as to populate them, both of these can be found on our public GitHub repository under the names `"physlog_table_creation.sql"` as well as `"physlog_populate_tables.sql"`.

Finally, see our model in full on the next page. The vector graphics should allow you to zoom in for better visuals.

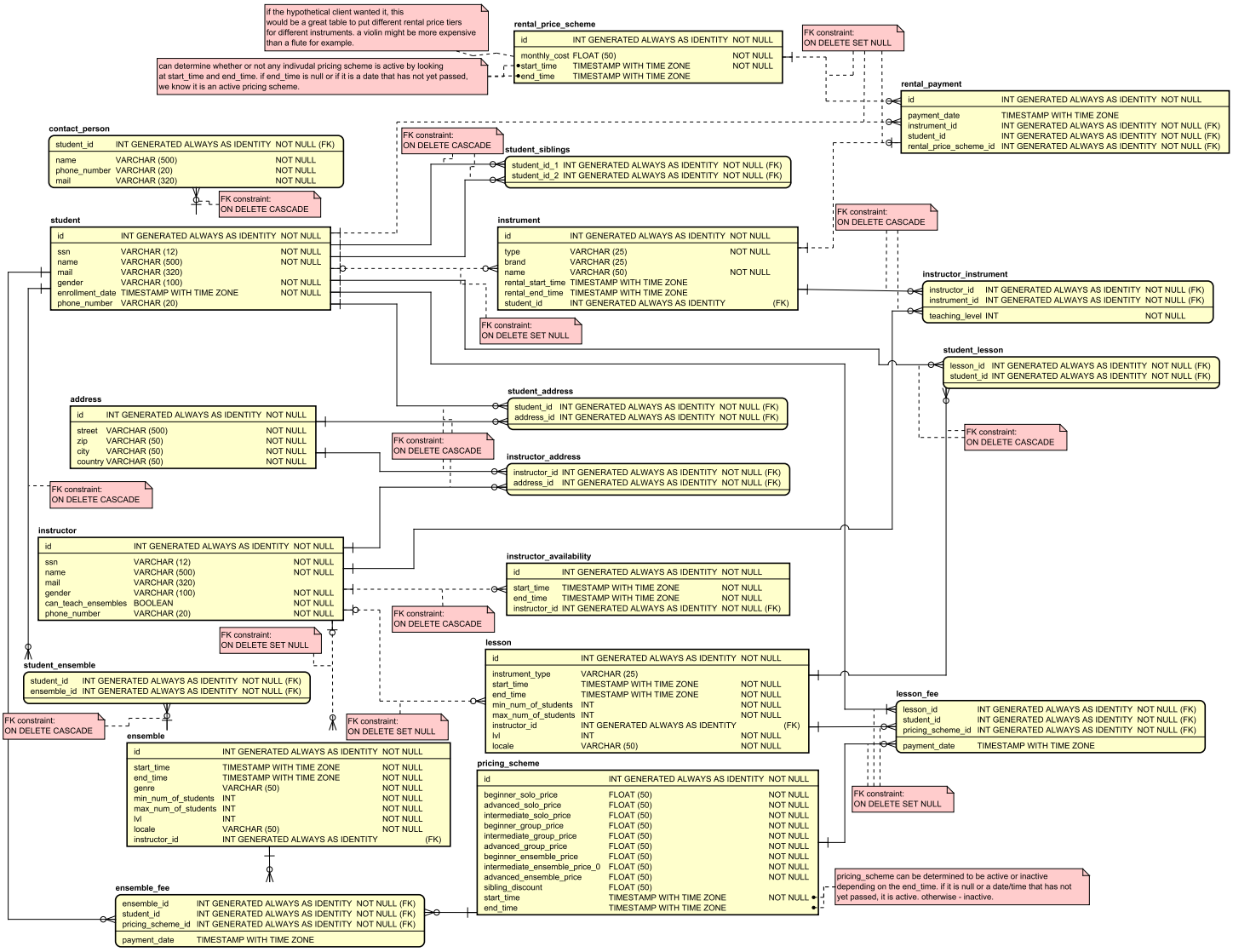

Figure 1: Our physical-logical model

5 Discussion

First and foremost, this assignment was incredibly challenging. While the group is generally satisfied with the work, we acknowledge that there is still a lot of room for improvement. One of the main weaknesses in our model is that we did not systematically convert all tables to 3NF; instead, we relied largely on intuition. It wasn't until the finalization stages, when we were nearly finished, that we properly analyzed the functional dependencies of all our columns. The normalization process was a new concept for us but despite our limited experience we managed to catch some of the biggest outliers.

For instance, we identified and corrected the "lesson" table, which originally included a "genre" attribute to distinguish ensemble lessons. In the final design, the table instead only represents solo and group lessons (the database distinguishes between the two by checking the minimum and maximum allowed students).

In hindsight, we recognize that attributes such as phone numbers, locales (teaching spaces) and the different pricing tiers could have been further normalized - with the clear downside that the diagram would become even more of a spiderweb of course. Converting a column such as a phone number for students or instructors into a look-up table would also give the developer the ability to store more than one phone number per person. Since the project description does not specify this need, however, we hope that our solution is fine as well.

It's easy to criticize your own work, and although some of our relations might not be properly in the 3NF state, they are at least highly readable and pragmatically built. For example, keeping all pricing tiers for different lessons in a single tuple of the "pricing scheme" table does make it significantly easier for developers who are using the database to get all relevant pricing data at once. Additionally, we imagine that there might not be more than a couple active pricing scheme tuples in the table at any given moment, as pricing schemes are typically set for extended periods of time before they expire (such as holiday offers) or change due to financial policy - which would mean that the need for normalization might not be as urgent.

The attributes "start_time" and "end_time" in the pricing scheme relations also allow us to determine the currently applicable pricing scheme for any given lesson, and as the pricing policy changes with time the old financial records will still carry a foreign key to the appropriate prices that were applicable at the time of transaction.

5.1 Self evaluation

1. **Are naming conventions followed? Are all names sufficiently explaining?**
   Answer: *We believe so. Words are separated with underscores and the names are descriptive of the data they represent, and all letters are uncapitalized.*

2. **Is the crow foot notation correctly followed?**
   Answer: *Yes. We believe so. It should accurately be reflecting if the relationships are 1:1, 1:N, or N:M. With "zero" included in the shape of a bubble.*

3. **Is the model in 3NF? If not, is there a good reason why not?**
   Answer: *This was touched upon in the discussion section above. We believe that the model is reduced to a decently efficient form - but that it could still be improved. We are new to the normalization process and are open to feedback on how to make it better suit 3NF.*

4. **Are all tables relevant? Is some table missing?**
   Answer: *Yes. We also don't believe that any table is missing, and that our model accurately represents the need of the Soundgood system.*

5. **Are there columns for all data that shall be stored? Are all relevant column constraints and foreign key constraints specified? Can all column types be motivated?**
   Answer: *Yes! We took great care to specify the types appropriately, and did not use numeric types for numbers that are not actually numeric such as phone numbers or social security numbers. Foreign key constraints are also specified using notes.*

6. **Are all relations relevant? Is some relation missing? Is the cardinality correct?**
   Answer: *To our understanding, all relations in the diagram are relevant and no key relation is missing.*

7. **Is it possible to perform all tasks listed in the project description?**
   Answer: *We've looked things over many a time and do believe that we've reached a state where all actions from the project description can be performed.*

8. **Are tables (or ENUMs) always used instead of free text for constants such as the skill levels (beginner, intermediate and advanced)?**
   Answer: *They are not. We are not sure if this is detrimental to our diagram or not since it was not mentioned the video lectures (that we can remember). We await the judgment of our peers and teachers to see if the lesson level or instrument skill levels ought to have been tables instead. In our solution, we represent them numerically as 1,2,3 for beginner, intermediate, advanced. In hindsight, this is something of a bad idea since they are not actually numeric so we ought not to represent them numerically.*

6 Comments About the Course

This report and task took significantly longer to complete than the first one - and we're unsure of how many hours we've accumulated. Some small comments might be that this task was incredibly challenging and maybe ought to have been split into two parts (logical model and physical model maybe) so that more time could have been spent practicing the normalization process for the logical model.

References

[1] Ramez Elmasri and Shamkant B. Navathe, *Fundamentals of Database Systems*, 7th ed., Pearson, 2016.

[2] Leif Lindbäck, *Logical and Physical Models, Part 1*, YouTube. Available 2024-11-18. URL: https://www.youtube.com/watch?v=yksFuRCXYqg

[3] Leif Lindbäck, *Logical and Physical Models, Part 2*, YouTube. Available 2024-11-18. URL: https://www.youtube.com/watch?v=GarIuyBqRmc

[4] Leif Lindbäck, *Logical and Physical Models, Part 3*, YouTube. Available 2024-11-18. URL: https://www.youtube.com/watch?v=wiQ7f3PYv6Q