Seminar 5 (2)
Data Storage Paradigms, IV1351

Dipsikha (Diddi) Dutta,
dipsikha@kth.se

5/1 2024

Contents

1 Introduction

The task of this seminar is to modulate a logical model with features of a physical model. The aim is to be able to create a normalized database with data specified for the requirements described for the description of *The Soundgood Music School*. I had little time left to do this assignment because of the other course, so I worked by myself, otherwise I would collaborate with some others. The ER diagram is made in the tool Astah and the requirements for the task is to have the diagram contained with all the different kinds of data needed for the business and to follow the guidelines of creating a logical model.

2 Literature Study

Before beginning with the assignment I went through the video material provided by the course and listed up all the important requirements for how a logical model should be modulated, and followed the rules taken up in the video material. I also read the article snippet “tips-and-tricks-task2” to prepare for this. Some important concepts described in those sources are about how to think when creating entities and attributes in those tables. It is also a huge part to identify which entities are the strong and which ones are the weak to be able to decide which cardinality relationship each entity would have. This leads to the next topic about normalizing the database which we are thinking about during the creation of the logical model and identifying the functional dependency of each attribute and deciding the primary and foreign keys. This is discussed more in the below sections.

3 Method

The method I used to proceed with this task is to follow the steps described in the video tutorials. Briefly described, the first tasks were about to include the attributes with one-to-one or one-to-many relationships in each entity and specifying the types of the attributes. Since we are using postgre databases most of the attributes were to be created as VARCHAR attributes. Then trying to decide the max length of each attribute value which I decided to be 50 which I consider is relatively high. After determining the constraints, NOT NULL and UNIQUE, the next step was to choose the relationships of each entity which are made to be student, instructor, lesson and person. The trickiest part of this assignment was to decide how the many to many relationship entities would be created. But I still followed the rule to export all the attributes with values higher than one, to be an entity of its own, which I still think is weird and makes it pretty complicated.

4 Result for seminar 5

https://github.com/Diddi25/Datalagring

The sql scripts and logical model can be downloaded from the map **sem5** in the github link above.

The feedback given from seminar 2 was:

1. Query for inserting data is not provided
It is now.
2. Most of the tables have a single attribute, it does not make any sense
I had difficulties to understand the meaning of many-to-many relationships which made me interpret the attributes in entity “booking” as such type of relation. This is however learned that it is not true and I have modified the model after having a deeper understanding of how the modeling works in the context of making SQL queries and normalizing the design.
3. what is the purpose of number of the table nr_students_in_lesson
It has the same reason as above, since a lesson can have many students I interpreted that number attribute as a many to many relationship but it is fixed now. I have changed the way to handle the nr of students in each lesson by having a min_students and max_students attribute in entity lesson.
4. naming convention is not followed properly, there is a pace between attribute names
They are modified to be have an underscore between the words even in the attributes now.
5. PK is not chosen wisely, for example table rental has a PK which is combination of ID+student_id+instrument_id, Only is sufficent in this case
In the case for rental entity the primary key is chosen to be rental id.

I made the model again from scratch this time since I was very unsure about these topics during seminar 2. The strong entities chosen in my new model is person, student, instructor, lesson, instrument, rental, skill and lesson_type. The strong entity person are having a relationship with the student and instructor since we need person nr, name, address and birth information about each of these students/instructors. The way to handle storing contact person details are explained below. The non-identifying relationships are used to not use foreign keys as composite primary keys in other entities.

Sibling
Since I now am sure about how to calculate the quantity of tuples in a table I dont have the attribute “quantity” in sibling entity like I had earlier. The primary key for

entity sibling is chosen as a composite key consisting of student id (FK) and the sibling id, since a student can have many siblings.

Contact details
The student can have many emails or many phone numbers. Implemented as a cross reference between these entities. (phone and email) The business description is specifying that contact details for a contact person is needed to be stored for each student. However, I chose not to store this contact person as a whole person with a person entity. The cross reference has the attribute is_contact_person as a boolean type, meaning if the phone or email belongs to the contact person or is the students contact details. Since the instructor does not have any contact person, an email and phone table is stored for each instructor. Meaning that the instructor only needs one work email and work phone number.

Lesson
Since a student can have many lessons and lessons can have many students there are made a cross reference student_lesson between the many-to-many relationship. The composite key in that entity is made of both the student id and lesson id since both primary keys are needed for this entity to exist. The lesson has more attributes than before since the operation of looking up available instructors that can give lessons can be looked up in another way than writing the available timeslots in a database. When booking a lesson there is usually a web application handled by the administration. The administration usually knows the schedules of the instructors, who have vacation or which lessons a specific instructor has by the start and end time tables in lesson, mapped to a specific instructor, which in further can decide which instructor can be available at the next lesson, and is further booked by the administration. This is why the booking-related entities in the previous model has been replaced with the attributes start_time and end_time. Since the group lessons depends on number of student enrollment the attributes max_student and min_student are created in the lesson entity instead of the nr_students_in_lesson entity as before.

Instrument:
To track the instruments used in a particular lesson there is a cross reference between the relations. Each instrument has a type, name, brand, quantity and price attribute as specified in the business description.

Rental:
The primary key is chosen to be rental_id. The entity consists of student and instrument id which is necessary information for the rental. Each rental has a start timestamp that has the constraint of not being null. This is because I do not believe a student has to specify when he wants to end his rental of the instrument. In the description is is described that each instrument has a lease period up to 12 months.

But the rental can be meant to be shorter or unknown at the start time of the rental, which is why end_rented can be null.

5 Discussion

- **Are naming conventions followed? Are all names sufficiently explained?**

Examples of the rules about naming conventions were to name the entities in lowercase and with underscore if there are more than one words in the entity name. This is followed. The names are given by its purpose to make it hopefully understandable.

- **Is the crow foot notation correctly followed?**

Yes, you can see that by the relationship symbols.

- **Is the model in 3NF? If not, is there a good reason why not?**

All non primary attributes in each entity, does not have a transitive relation where the attribute is not dependent on its primary key.

One exception is the instrument type specified for each lesson, the lesson_intrument_types entity. In the conceptual model this entity was at first an attribute in the lesson entity. Since instrument types can be different for each lesson_type, for example there can be several instrument types in an ensemble lesson and only one type for group and individual lessons, this entity is made. On the other hand we can track the instrument used for each lesson by the students' rentals of instruments. But this would not have worked in the case if the student has not rented any instruments but only borrows instruments for each lesson from the school's instrument stock.

- **Are all tables relevant? Is some table missing?**

In order to be approved to the school, the administration has to be able to see how many students and teachers there are available and if there is any place. A student also have to be approved for the bookings it makes for participating in lessons, which are also handled by the administrative staff which

- **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?**

Almost all the entities have the constraint NOT NULL because they would have a higher cardinality than one if the entity relation is relevant. For example the student can have one to many lessons_taken_per_month, which makes the nr of lessons attribute and lesson type attribute NOT NULL if there are at least one lesson taken.

The quantity of siblings in the entity nr_of_siblings can be NULL because all the students don't have a sibling that is active in school. Because of the eligibility of getting a discount by the number of siblings and its participation in school, each sibling has their UNIQUE sibling id. And can be related to one other student, which is identified with its own student id.

Since there is only one address that is relevant in the business description, the address(es) of each person, the address entity´s attributes are separated from the person entity.

- **Can the choice of primary keys be motivated? Are primary keys unique?**

No entity has a foreign key as primary key which was motivated to be incorrect in the tutorial. The composite foreign primary keys makes each entity primary key unique.

- **Are all relations relevant? Is some relation missing? Is the cardinality correct?**

The price entity which collects all the price attributes is mainly in relationship with the booked_lessons_student entity. It could be argued that this entity would have a relationship to lessons_given_from_teacher but the reason for not making the diagram too messy, the teacher payment can be calculated by the number of bookings for each student.

- **Is it possible to perform all tasks listed in the project description?**

Operations:

- To be approved to the school:

To be approved to have a place in the music school the administration would like to know the total number of teachers employed at the school which can be calculated by the number of instructor id:s. If there are enough places to have students in the school the total number of students would be calculated, which can be resolved in the same way.

- To book a lesson:

The student skill level, which teachers, locations and time slots are available can be queried by the tables accordingly. The details about each lesson can be tracked with its lesson type, skill level, which teacher instructed, the nr of students participated in that lesson and the location. The genre can be null if the lesson type is of “group” or “individual”, which however goes against the guideline of having as few null elements as possible in a database, but it is as it is now.

- Rental:

The nr of instruments each student can rent up to are a business rule that is specified as a note. The rental id can be tracked which gives all the details about the rental, which instrument, its id, to which student by the students id. The lease period

(for how long the instrument can be rented) and the rented time (the time the student has been rented one instrument) can be recorded.

- Fees and payments:

The data related to the number of lessons taken from each student per month can be tracked with the number of lessons attributes recorded for each student id. The payment for each instructor can be calculated in the same way with the composite foreign keys.

- **Are all business rules and constraints that are not visible in the diagram explained in plain text?**

Yes, if foreign keys are to be deleted they are deleted on cascade automatically. The business rules are also explained in pink notes.

- **Are there attributes which are calculated from other attributes and then written back to the database (derived attributes)? If so, why? Records of student fees and instructor payments might be examples of such attributes.**

There are no entities such as student fees or teacher payment either in the logical model nor in the database scripts. However, since it is specified in the business description that the administration would like to be flexible with change of each price attribute, there have been different entities for each type of price. Since all these attributes can have values with higher cardinality than one, they are made as entities.

- **Are tables (or ENUMs) always used instead of free text for constants such as the skill levels (beginner, intermediate and advanced)?**

The lesson_type entity is supposed to illustrate the enum types which can be beginner, intermediate and advanced. I followed the example in the text tutorial snippet in “tips and tricks for task2”.

- **Is the method and result explained in the report? Is there a discussion? Is the discussion relevant?**

Yes