# Project Report

**Data Storage Paradigms, IV1351**

Your Name and Email Address

Date

## 1 Introduction

This is the second task of the project where a logical/physical model should be created. The same requirements of the Sound good music school should be applied as it was in the previous task. The purpose is also to create a database and fill it with proper data.*In this task, I have worked with Ruth Jenbere Shewa and Lassya Desu.*

## 2 Literature Study

As always there were some critical steps to understand in order to create the model. That information exists in several videos that the teacher had prepared. Therefore, they were watched carefully. Also there were other lectures and PDF files as a help to better solve the task and avoid mistakes. For example, I have learned that the model should only provide data and that I shouldn't think about the how, meaning how the information should be handled. That part is what the coding does. The requirement is only that relevant information and calculations can be done based on the proper information that exists in the created model.

## 3 Method

To solve this task, the teacher's uploaded videos were followed consisting of eleven steps. First, a table was made for every entity from the conceptual model. Second, a column was made for every attribute with a cardinality 0...1 or 1...1. Attributes with a higher cardinality got placed in each of their own tables. Fourth, the type of every attribute was decided, whether it was a char, varchar, int and so on. Fifth, the column constraints were determined, meaning whether the attribute should be NOT NULL, UNIQUE or both. Sixth, primary keys were assigned to the strong entities. Surrogate keys were created and applied in most cases. The seventh step was to identify all the one-to-one and one-to-many relations and create them. The PK of the strong end was used as a FK (foreign key) in the weak end. If the weak end had a meaning without the strong end, a

surrogate PK was made for it, otherwise, the FK was used as a PK. The FK-constraints should then be decided. There were three options: "prohibit" where no action is done, "allow" where the attribute sets to null and "delete" where a cascade operation is done meaning if the parent entity is deleted the child entity is also deleted. Eighth, cross-reference tables were made for each many-to-many relation. The PKs of the two entities are FKs in the cross-reference table. The PK of the table should be the FKs combined. FK constraints are considered and decided by writing notes. It is also possible to add more attributes to this table. The relations from the to entities to this table must also be correct. Ninth, the multi-valued attributes are handled. FKs are assigned to them from the relevant entities and their PK is the FK and the multi-valued attribute combined. Tenth, it should be verified that the model is normalized. And eleventh, it should be verified if it is possible to perform all planned operations on the data.

# 4 Result

The tables made in the first place were person, student, instructor, instrument, rental, contact person, lesson and pricing scheme. Columns were made for the attributes with cardinality 0...1 and 1...1. "name" in "lesson" is and example of such an attribute. Multi-valued attributes were place in separate tables which were telephone_number, e_mail, available_time, instrument_type, fixed_slot, appointment and desire_instrument_type. In this, step the type of the attributes were decided. For example, attributes such as name, person number, skill level and genre were assigned the type varchar with a limit of fifty characters. Surrogate keys were assigned int GENERATED ALWAYS AS IDENTITY to always have a unique value in the database. An FK assigned from an entity with the surrogate key of a type int Generated always as identity should only have the type int according to the instructions. For example, the Fk from the person in the instructor is only int, while the surrogate key of the person is int GENERATED ALWAYS AS IDENTITY. There are many attributes that cannot be Null, therefore the constraint NOT NULL is applied. Some examples are the ids, name, person number, and other contact details according to the requirements.

Strong entities such as person, lesson, student, instructor, rental, pricing scheme and instrument have all surrogate keys as primary keys since they should be totally unique. Person is a strong entity, therefore FKs were assigned from this table to student, instructor and contact person so that they can have access to all the attributes in the person table. Also, each of them is counted as a person. Since these table have a meaning on their own, surrogate keys were assigned to them as their primary keys. The name conventions are followed. For example, in this case the FKs are called person_id since they are assigned from the person entity. The FK constraint for all of the FKs was decided as ON DELETE CASCADE which is indicated through a note in the model. For example, a student doesn't exist without a person(firgure 1).

There are two cross-reference tables in this model. The first one is between lesson and student, since they had a many-to-many relation. FKs are assigned to this table from both student and lesson as columns. An instance of lesson should have at least one

student up to many student. An instance of student can on the other hand have zero or many lessons. The PK is the two FKs combined to make a unique combination of every student with a lesson. Another cross-reference table was made for the student's many-to-many relation to itself, the relation indicating a sibling relation.

Multi-valued attributes are assigned FKs from the relevant entities. For example, person assigns FKs to telephone number and e-mail. Both the FK and the multi-valued attribute build a composite primary key for the sake of uniqueness. An instance of person can have zero to many e-mails but an email can only be assigned to one person.

Link to the repository: https://github.com/ruthshewa/SoundGoodMusicSchool

# 5 Discussion

Naming conventions are followed as table names and attribute names are descriptive. For example, the instrument type that the student desires to learn is called desired_instrument_type which is a multi-valued attribute, stored in its own table.

The crow-foot notation is correctly followed. For instance, a person can have zero to many telephone numbers, while a telephone number can only belong to at least one person. All of the primary keys are unique. Either it is a surrogate key with the type int GENERATED ALWAYS AS IDENTITY which makes the database generate a unique value for every row, or it is a composite primary key made of a surrogate key and a foreign key. For example, person, lesson, rental, student and instructor have PKs that are surrogate keys of the type int GENERATED ALWAYS AS IDENTITY. The multi-valued attributes e-mail, telephone number, available time and instrument type have composite PKs.

There aren't any attributes than can be calculated by other attributes. For example, there was a table called payment, where the salary of an instructor and the amount that a student should pay were attributes. These are counted as derived data, since they can be calculated by other attributes. For instance, the lesson table which is related to the student and instructor will provide the information of how many lessons a student has taken and the instructor has taught. Therefore, no table as payment is needed. So it was deleted.

# 6 Comments About the Course

There was so much required for this task and so little information. Specially when creating the database and the generated data. There was absolutely no information on how to do the generation and what to think about when doing so. Maybe uploading some video tutorials could help the future students.

Tutorials are only a few hours and only two times which are clearly not enough. Also, only one Lab assistant available which makes one need to wait a very long time before getting help.

# 7 Changes after the feedback

To fix bullet one, the attribute available_quantity in instrument was changed to available to address whether a specific instrument is available or not. Since instrument_id and student_id are foreign keys in the rental entity, it is possible to track which instrument is rented by who. If its not rented at all, it is indicated through the "available" attribute in instrument(figure 2).

To handle bullet two, the "price" attribute was moved from rental to instrument. Here, the price is called "rental_price" and indicates the cost to rent a specific instrument.

Bullet three was fixed by adding an attribute called "lesson_type" to the pricing_scheme. This way, it is clear, which lesson type a specific price belongs to.

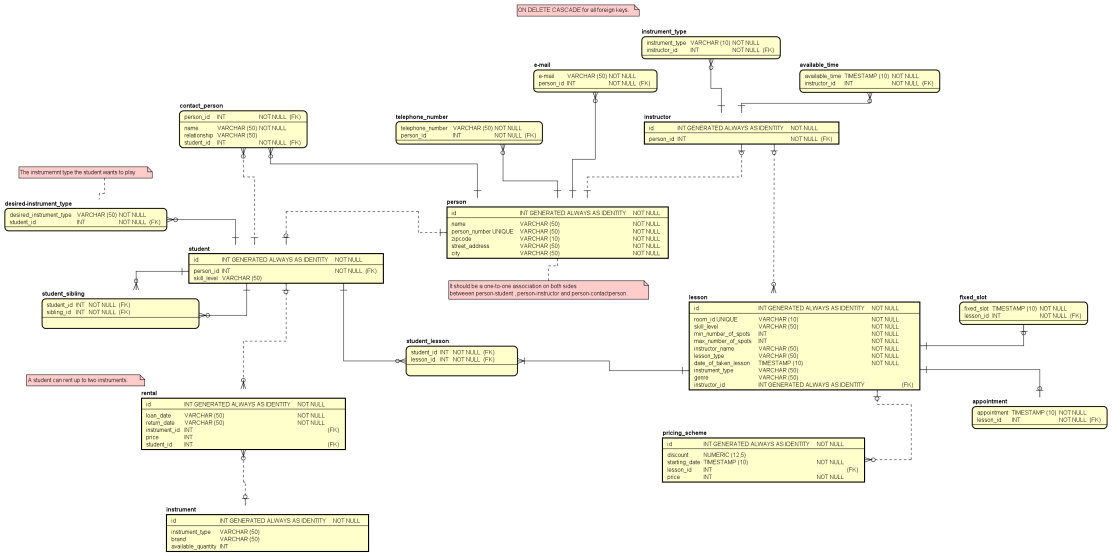


Figure 1: Logical model

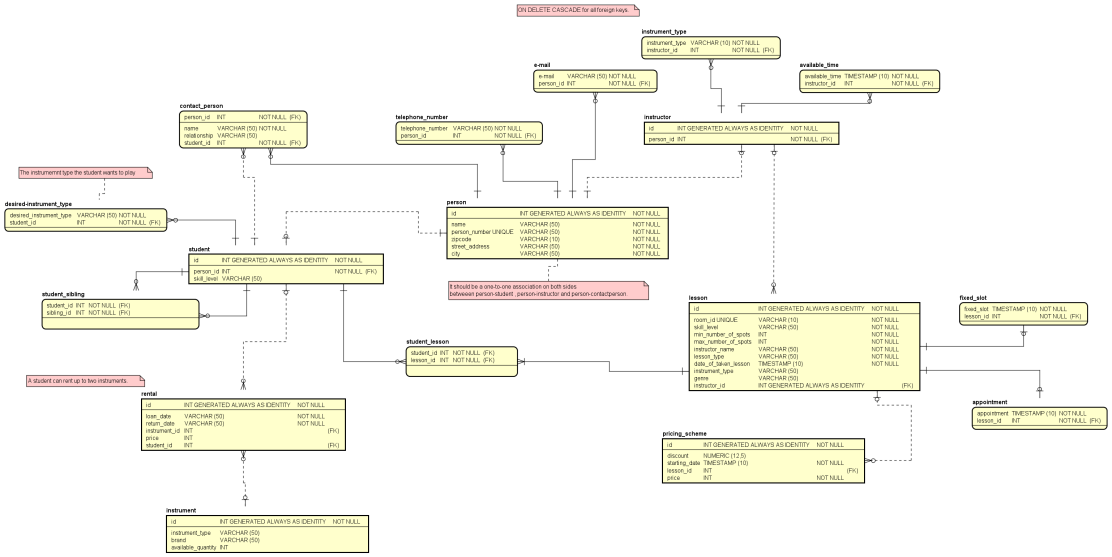


Figure 2: logical model updated version